A GENERALIZED PROCESS FOR THE VERIFICATION AND VALIDATION OF MODELS AND SIMULATION RESULTS

submitted by

Dirk Brade

DISSERTATION

for the achievement of the academic degree

doctor rerum naturalium (Dr. rer. nat.)

at the

Fakultät für Informatik

Universität der Bundeswehr München

Supervisors:

Prof. Dr. A. Lehmann

Prof. Dr. R. K. Huber

Neubiberg, October 2003

Universität der Bundeswehr München
Fakultät für Informatik

Thema der Dissertation: A Generalized Process for the Verification and Validation of Models and Simulation Results

Verfasser: Dirk Brade

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Gunnar Teege
1. Berichterstatter:	Prof. Dr. Axel Lehmann
2. Berichterstatter:	Prof. Dr. Reiner K. Huber
Weiterer Prüfer:	Prof. Dr. Fritz Lehmann
Weiterer Prüfer:	Prof. Dr. Mark Minas

Tag der Prüfung: Montag, 23. Februar 2004

Mit der Promotion erlangter akademischer Grad: Doktor der Naturwissenschaften

(Dr. rer. nat.)

Neubiberg, den 10. März 2004

"A pessimist sees the difficulty in every opportunity;
an optimist sees the opportunity in every difficulty."

Sir Winston Churchill

ACKNOWLEDGEMENTS

During the development of the work presented in this document, I got invaluable support from numerous individuals. I would like to express my gratitude to all these persons, although it is impossible for me to mention all of them by name. However, the following persons contributed to the creation of the thesis in such a manner that their names need to be revealed in this place:

I would like to thank Prof. Dr. Axel Lehmann, who directed my research over the years and opened any door I needed or liked to pass through. I owe thanks to him for the great opportunities he created for me, and especially the strong support for presenting research results to the international Modeling and Simulation community. Further on I thank Prof. Dr. Reiner K. Huber for co-supervising this thesis work.

I thank my fiancée Katja for her encouragement and care, and her permanent motivation to expand my horizon. When I was tired of working, you inspired me to resume. Special thanks go to my parents and my brother – I highly appreciate everything you did for me during the last three decades. Without the foundations laid by you, the completion of this thesis would have been impossible.

Numerous persons provided valuable thoughts and ideas for my work, among them Michelle Kilikauskas (who also gave extremely helpful advice on how to convert my German-English at least into American-English), Richard Maguire, Simone Youngblood, Johannes Lüthi, Marko Hofmann, and Scott Harmon. Stephan Mros facilitated the development of the sample M&S project by providing the required real world data.

The project work for the institute was challenging and sometimes stressful, but great colleagues turned working hours into a great time – thanks to all of you. Carmen Waldner did an excellent job with the implementation of the tool VIOLADE, and Siegfried Pohl showed admirable patience while looking through this document on the search for errors.

Finally, I also owe thanks to all those I did not mention here, for countless interesting and sometimes hot discussions, their criticism and help, their readiness to dispute and their interest in my work.

Dirk Brade, Oktober 2003

ABSTRACT

With technologies increasing rapidly, symbolic, quantitative modeling and computer-based simulation (M&S) have become affordable and easy-to-apply tools in numerous application areas as, e.g., supply chain management, pilot training, car safety improvement, design of industrial buildings, or theater-level war gaming. M&S help to reduce the resources required for many types of projects, accelerate the development of technical systems, and enable the control and management of systems of high complexity. However, as the impact of M&S on the real world grows, the danger of adverse effects of erroneous or unsuitable models or simulation results also increases. These effects may range from the delayed delivery of an item ordered by mail to hundreds of avoidable casualties caused by the simulation-based acquisition (SBA) of a malfunctioning communication system for rescue teams. In order to benefit from advancing M&S, countermeasures against M&S disadvantages and drawbacks must be taken. Verification and Validation (V&V) of models and simulation results are intended to ensure that only correct and suitable models and simulation results are used. However, during the development of any technical system including models for simulation, numerous errors may occur. The later they are detected, and the further they have propagated through the model development process, the more resources they require to correct – thus, their propagation should be avoided. If the errors remain undetected, and major decisions are based on incorrect or unsuitable models or simulation results, no benefit is gained from M&S, but a disadvantage.

This thesis proposes a structured and rigorous approach to support the verification and validation of models and simulation results by

- the identification of the most significant of the current deficiencies of model development (design and implementation) and use, including the need for more meaningful model documentation and the lack of quality assurance (QA) as an integral part of the model development process;
- giving an overview of current quality assurance measures in M&S and in related areas. The transferability of concepts like the capability maturity model for software (SW-CMM) and the ISO9000 standard is discussed, and potentials and limits of documents such as the VV&A Recommended Practices Guide of the US Defense Modeling and Simulation Office are identified;
- analysis of quality assurance measures and so called V&V techniques for similarities and differences, to amplify their strengths and to reduce their weaknesses.
- identification and discussion of influences that drive the required rigor and intensity of V&V measures (risk involved in using models and simulation results) on the one hand, and that limit the maximum reliability of V&V activities (knowledge about both the real system and the model) on the other.

This finally leads to the specification of a generalized V&V process – the V&V Triangle. It illustrates the dependencies between numerous V&V objectives, which are derived from spe-

cific potential errors that occur during model development, and provides guidance for achieving these objectives by the association of V&V techniques, required input, and evidence made available. The V&V Triangle is applied to an M&S sample project, and the lessons learned from evaluating the results lead to the formulation of future research objectives in M&S V&V.

ACRONYMS

AAA	American Automobile Association
ADP	An der Point
API	Application Programmers Interface
BAB	German: Bundesautobahn (Interstate)
CAD	Computer Aided Design
CAE	Claim-Argument-Evidence
CLIMB	Confidence Levels In Model Behavior
CM	Conceptual Model
CTL	Computation Tree Logic
DERA	U.K. Defence Evaluation and Research Agency
DEVS	Discrete Event Simulation System
DMSO	Defense M&S Office
DOD	US Department of Defense
DOE	Design Of Experiments
EM	Executable Model
FEDEP	Federation Development and Execution Process
FM	Formal Model
FOM	Federation Object Model
HIL	Hardware in the Loop
HLA	High Level Architecture
HW	Hardware
I/O	Input/Output
I/S/O	Input/State/Output
IP	Intermediate Product
IT	Information Technologies
ITOP	International Test Operations Procedure
IV&V	Independent Verification and Validation
KM	German: Konfigurationsmanagement (Configuration Management)
M&S	M&S
MIL	Man in the Loop
N/A	Not Applicable
ODBC	Open Database Connectivity
OMT	Object Model Template
ORB	Object Request Broker
PLTL	Propositional Linear Temporal Logic
PM	German: Projektmanagement (Project Management)
PN	Petri Nets
QA	Quality Assurance
QN	Queuing Nets
QS	German: Qualitätssicherung (Quality Assurance)

RAND	Contraction of the term “research and development”
RNG	Random Number Generator
RPG	Recommended Practices Guide
RTI	Runtime Infrastructure
RTTL	Real Time Temporal Logic
SBA	Simulation-based Acquisition
SE	German: Systementwicklung (System Development)
SIL	Software In the Loop
SME	Subject Matter Expert
SMV	Symbolic Model Verifier
SN	Sponsor Needs
SOM	Simulation Object Model
SPD	Structured Problem Description
SR	Simulation Results
SRT	Sweeney, Robertson, and Tocher
SW	Software
TL	Temporal Logic
UML	Unified Modeling Language
V&V	Verification and Validation
VV&A	Verification, Validation, and Accreditation
VV&T	Verification, Validation, and Testing
VW	Volkswagen
xIL	x (man, machine, or software) In the Loop
XML	Extensible Markup Language

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **VII**

ABSTRACT **IX**

ACRONYMS **XI**

TABLE OF CONTENTS **XIII**

1 INTRODUCTION **1**

1.1 The Role of System Theory in M&S **2**

1.2 Modeling and Models **3**
1.2.1 Physical vs. Symbolic Models 4
1.2.2 Symbolic Description of Structure and Behavior 4
1.2.3 Hierarchical Decomposition 4
1.2.4 Continuous and Discrete Models 5
1.2.5 Stochastic and Deterministic Models 5
1.2.6 Representation Forms of a Model 5

1.3 Model Solving **6**
1.3.1 Analytic Solution Methods 7
1.3.2 Computer-Based Simulation 7

1.4 Model Development and Execution Process **8**
1.4.1 Modeling Activities and Intermediate Products 8
1.4.2 “Waterfall” Processes 10
1.4.3 V-Type Processes 10
1.4.4 Life Cycle Processes 10
1.4.5 Spiral Processes 10
1.4.6 The Chosen Model Development and Execution Process 11
1.4.7 Parties Involved 12

1.5 Verification and Validation of Models and Simulation Results **13**
1.5.1 Correctness and Verification 14
1.5.2 Suitability and Validation 15
1.5.3 Demarcation: Software Quality Assurance 16
1.5.4 Credibility 17

1.5.5 Accreditation 17

2 GOALS OF THIS THESIS 19

2.1 Current Deficiencies in V&V and Challenges in the Problem Field 19
2.1.1 Model Development Considered as Software Development 20
2.1.2 Missing Model Documentation 20
2.1.3 Under-Appreciation of Constructive Aspects of V&V 21
2.1.4 Missing V&V Guidance and Lack of Comparability 21
2.1.5 Missing Estimate of Required V&V 21
2.1.6 Missing Documentation of V&V Activities Previously Conducted 22

2.2 Focus of the Thesis 22

3 THE STATE OF THE ART 25

3.1 Principles of V&V Planning and Implementation 25
3.1.1 V&V as Integral Part of Model Development 25
3.1.2 Precise Specification of the Intended Purpose 25
3.1.3 Sufficient System Knowledge 26
3.1.4 Sufficiently Unbiased V&V Implementers 26
3.1.5 Minimal Degree of Subjectivity 26
3.1.6 Unfeasible Proof of Suitability and Correctness 27

3.2 Existing Related Standards, Guidelines, Methods, and Procedures 27
3.2.1 The Capability Maturity Model for Software 27
3.2.2 ISO 9000 29
3.2.3 The V-Model 30
3.2.4 NASA Likelihood of Failure Rating 31
3.2.5 The DMSO Recommended Practices Guide on VV&A 31
3.2.6 Confidence Levels in Model Behavior 32
3.2.7 RAND VV&A Taxonomy 34
3.2.8 DERA Risk Classification 36
3.2.9 Balci’s Evaluation Environment 36
3.2.10 The International Test Operations Procedure on V&V 37

3.3 Summary of Basic Concepts 38
3.3.1 Process Approach 38
3.3.2 Question Catalogues and Fixed Indicator Hierarchies 38
3.3.3 Flexible Indicator Hierarchies 39
3.3.4 Risk-Orientation 39
3.3.5 Process Assessment vs. Product Assessment 39
3.3.6 V&V Degrees or V&V Levels 40

4 THE GENERALIZED V&V-PROCESS 41

4.1 Application Risk and Required V&V Effort 42
4.1.1 General Approach to Risk Assessment 42
4.1.2 Required V&V Rigor and Intensity 44
4.1.3 Risk and Credibility Classification 44

4.2 Real System Knowledge 45
4.2.1 System Structure 46
4.2.2 System Behavior 46

4.3 Model Knowledge 47
4.3.1 The Three-dimensional Model Information Space 48
4.3.2 Populating the Three-Dimensional Model Information Space 50

4.4 Establishing Credibility 51
4.4.1 Credibility of the Model 52
4.4.2 Credibility of Runtime Input Data and Embedded Data 53
4.4.3 Credibility of Model Operation 55

4.5 Remarks on V&V Techniques 55
4.5.1 Demarcation of Validation, Verification, Testing, and Analysis 56
4.5.2 Expert Opinion as Validation Evidence 58

4.6 The Dependencies Visualized and Roles in V&V 58

4.7 Structured V&V – The V&V Triangle 59
4.7.1 Intermediate Products 62
4.7.2 Failures, Errors, and their Propagation 62
4.7.3 V&V Main Phases 64
4.7.4 V&V Sub-Phases and V&V Objectives 65
4.7.5 Incremental Credibility Building and Tailoring 66

5 EVALUATION OF THE V&V TRIANGLE 71

5.1 The Sample M&S Project 71

5.2 Documentation Correctness in Form and Internal Consistency 72
5.2.1 Background – Structured Documents 72
5.2.2 The Enablers: Check Lists, Templates, Dependencies Matrices, and Formalisms 73
5.2.3 Planning with the V&V Triangle 75
5.2.4 Sub-Phase 1.1: Correctness in Form and Self-Consistency 76
5.2.5 Sub-Phase 2.1: Correctness in Form and Self-Consistency 77
5.2.6 Sub-Phase 3.1: Correctness in Form and Self-Consistency 78
5.2.7 Sub-Phase 4.1: Correctness in Form and Self-Consistency 79
5.2.8 Sub-Phase 5.1: Correctness in Form and Self-Consistency 80
5.2.9 Concluding Remarks 80

5.3 Automatic Detection of Behavior Proposition Violation 81
5.3.1 Background – Formal Specification of Software Requirements 81
5.3.2 The Enabler: Temporal Logic 82
5.3.3 Planning with the V&V Triangle 83
5.3.4 Sub-Phase 2.1: Extracting Behavior Propositions from the Conceptual Model 85
5.3.5 Sub-Phase 2.2: Counterchecking with the Structured Problem Description 86
5.3.6 Sub-Phase 2.3: Counterchecking with the Sponsor Needs 86
5.3.7 Sub-Phase 3.2: Model Checking 87
5.3.8 Sub-Phase 4.3: Automated I/S/O Trace Analysis 88
5.3.9 Sub-Phase 5.4: I/S/O Trace Surveillance 89

5.3.10 Concluding Remarks on PLTL for Behavior Propositions Specification 90

5.4 Human Review 90
5.4.1 Background – Current Practice 91
5.4.2 Enabler: Human Knowledge and Flexibility 91
5.4.3 Planning with the V&V Triangle 92
5.4.4 Sub-Phase 1.1: Problem Description Review 93
5.4.5 Sub-Phase 2.1 and 2.2: Symbolic Description Review and Mental Execution 93
5.4.6 Sub-Phase 4.1, 4.3, and 4.4: Visualization by Animation 94
5.4.7 Sub-Phase 5.1: Results Assessment 96
5.4.8 Sub-Phase 5.4: Requirements Satisfaction Assessment 96
5.4.9 Concluding Remarks on Human Review 96

6 SUMMARY AND CONTINUING WORK 97

6.1 Contribution to the State of the Art 97

6.2 The V&V Triangle 98

6.3 Potential Benefit and Limits of the V&V Triangle 98
6.3.1 Potential Benefit 99
6.3.2 Limits 99

6.4 Application V&V Techniques 100
6.4.1 Requirements Tracing and Configuration Control 100
6.4.2 Automated Error Detection 100
6.4.3 Instrumentation, Visualization, and Face Validation 100

6.5 Other Promising Paths Through the V&V Triangle 101
6.5.1 Instrumentation, Assertions, and Inductive Assertions 101
6.5.2 Statistical Data Evaluation 101
6.5.3 Flow Analysis 102

6.6 Concluding Remarks on the V&V Triangle 102

6.7 V&V of Component-Based Models 103

6.8 V&V of Distributed Simulations 104

6.9 Assessment of Credibility 104

6.10 Tool Support 104
6.10.1 Model Documentation System 105
6.10.2 An Integrated Model Development System 105
6.10.3 Output Analyzers 105

REFERENCES 107

INDEX OF FIGURES AND TABLES 115

APPENDICES 117

1 INTRODUCTION

The importance of modeling and simulation (M&S) is increasing constantly in numerous application areas, including industry, academia, the military, and even public life. This cost-effective and time-saving technology is used to gain new knowledge about a real system and its behavior, without requiring the real system actually to exist or to be manipulated. With the growing potential of M&S, the impact of decisions directly or indirectly based on simulation also grows – and this increases the possible worst case impact of wrong or invalid models or simulation results on the real world.

There is no doubt that attention is required when relying on symbolic, quantitative models and computer-based simulation. Although the potential for their application has not yet been fully explored, non-negligible damage has already been caused by computer-based models and failures in control software, as the following examples illustrate:

- *The Grounding of Royal Majesty*: In 1995, the Royal Majesty, a three year old ship equipped with a modern navigation system, strayed 17 miles off course and ran aground, only ten miles away from the US coast [Frankfurter Allgemeine Zeitung 1995]. Due to fortunate circumstances the ship only grounded on mud and nobody was killed. Malfunction of the GPS antenna caused the navigation system to switch into backup mode, which used a dead reckoning model for position estimation. However, the effects of wind and current on the position of the ship were not taken into account when modeling the ship's movement, yielding an invalid model and wrong, nearly fatal position data from dead reckoning [Epstein 1997].
- *The Pentium Bug*: In 1994 an error in the floating point division of Intel's Pentium Processor was detected. To accelerate floating point division, the Pentium uses a speedy algorithm known as Simulation ResultsT division, which requires a look-up table for updating the remainder [Peterson 1997]. Although the table entries were determined correctly, a bug in the transfer script distorted the table that was finally used for the design of the processor. Five wrong entries in a table with 1066 cells caused Intel an approximated loss of half a billion US-Dollars. Intel did not publish where exactly in the design process the error occurred. However, because an error pattern can be detected [Edelman 1997], it must be assumed that the transfer from the look-up table to the design model of the integrated circuit failed, and the error was propagated through layout and production. Although M&S are used intensively for integrated circuit design and development, the error was not detected prior to the release of the processor.
- *Delay of the Denver Airport Baggage System*: According to [General Accounting Office 1994], in 1994 significant mechanical and software problems plagued the automated baggage handling system of the new Denver airport. In tests of the system, bags were misloaded, were misrouted, or fell out of telecarts, causing the entire system to jam. Numerous mechanical deficiencies and software errors caused the undesired behavior; however, modeling errors were also identified. For example, the baggage sys-

tem loaded bags into telecarts that were already full. After a previous jam the system was restarted, but initiated incorrectly, which led to an incorrect reflection of the state of the real system in the model used by the control software. The estimated overrun through February 1995 given in the report is US-$360 million, plus US-$37 million loss in income that the airport would have generated, if opened one year earlier, plus US-$86 million for the alternative baggage transportation system. Unconfirmed sources estimate a final overrun to date of US-$4.2 billion.

This list can be easily continued. Reasons for the above failures are numerous, among them underestimation of system complexity, time pressure, and budget limitations. But aside from these very practical problems that more or less influence the design and development of any technical system, M&S lack a commonly accepted (and applied) methodology for quality control. This situation is not acceptable when the intended use of a model or simulation results has been deemed safety critical.

To further facilitate the successful application of M&S in the military, in 1995 the US Department of Defense published the M&S Master Plan, which describes the M&S objectives of the US Armed Forces [Department of Defense 1995]. To fully explore the benefits of M&S and to avoid their pitfalls, the necessity of the development of a verification and validation methodology for models and simulations was included in the master plan, and explicitly stated within Instruction 5000.61 [Department of Defense 1996]. Subsequently this necessity was also recognized by the NATO M&S community and included in the NATO M&S Master Plan [North Atlantic Treaty Organization 1998]. The need for models and simulations within the German Federal Armed Forces has been formulated in the “Leitlinie für Modellbildung und Simulation in der Bundeswehr“ [Bundesministerium für Verteidigung 2000] in 2000. The desired use of M&S for simulation-based acquisition (SBA) is stated in [Bundesamt für Wehrtechnik und Beschaffung 2000].

For the use of M&S in industry, no master plans or official statements concerning verification and validation have been formulated and presented to the public so far. However, the increasing time-to-market requirements imply that there also is a need for methods that enhance the availability of time and cost saving credible models and simulation results.

This thesis is about verification and validation of symbolic models and results of computer-based simulation, as will be motivated and developed in more detail in chapter 2. In the remainder of this chapter, those concepts of M&S relevant to the context of this document are briefly reviewed to create a common base of understanding and to enhance the understanding of the work presented here. Then the terminology of verification and validation is introduced. This discussion cannot replace a complete introduction to the particular topics, and is not intended to do so.

1.1 The Role of System Theory in M&S

In the real world numerous processes and activities take place that are determined by the entities or objects involved, their properties, their behavior, their dependencies, and their relationships. These processes or activities will create new objects or entities, changes of properties, new behaviors, new dependencies, new relationships or new processes. Often this construct of entities, relationships, causes, and effects is so complex that the human mind needs to idealize and abstract in order to understand the real world. Among the aims of system theory is to enhance the understanding and mastery of complex systems.

[Simon 1962] follows the “divide and conquer” approach and defines a complex system as “one made up of a large number of parts that interact in a non-simple way. In such systems the whole is more than the sum of its parts, not in an ultimate, metaphysical sense, but in the important pragmatic sense that, given the properties of the parts and the law of their interaction, it is not a trivial matter to infer the properties of the whole”. In system theory the real

world is perceived as an unlimited set of parts (entities or objects), which somehow interact with each other.
System theory distinguishes between *system structure* and *system behavior* [Zeigler, Praehofer, and Kim 2000]. To handle system structure and to order the parts, objects, or entities, a *hierarchical structure* within the system is assumed. [Simon 1962] defines it as "a system that is composed of interrelated subsystems, each of the latter being in turn hierarchic in structure until we reach some lowest level of elementary subsystem". To get a grip on system behavior he distinguishes between "the interaction *among* subsystems, on the one hand, and interactions *within* subsystems – this is among the parts of those subsystems – on the other."
This concept of *hierarchical decomposition* was adopted for M&S. The structural breakdown of a complex system allows to a certain degree the independent analysis of its subsystems. This depends on the *decomposability* of the system. If the interaction among subsystems is of the same magnitude as the interactions within the subsystems, the system is *not decomposable*, if there are no interactions, the system is *decomposable*. Complete decomposability of a system is rare, therefore the term *near-decomposability* is introduced [Courtois 1981]. (Near-decomposability means that the interactions among subsystems are weak, but not negligible.)
To avoid confusion, in the following text the terms *reality* and *real world* will always refer to the world we live in, which the author considers to be real. A cutout (set of entities) from the real world then is called a *system* and is referred to as *real system* in the following, with a *system border* between the system of interest and the rest of reality.

1.2 Modeling and Models

Both M&S have shown themselves to be extremely useful for understanding and managing complex real systems. However, a new age of M&S has begun with the rapid innovation in computer science and technology and its direct impact on M&S theory and practice. This section clarifies the difference between *modeling* and *simulation*, and motivates the selection of the thesis focus on quantitative, symbolic models and computer-based simulation.
Whenever one needs to communicate knowledge about a real system (*teach*), to understand the structure of and the dependencies within the real system (*analyze*), to estimate further development of a given situation (*prognosticate*), or to create a new system (*synthesis*), one will be confronted with the complexity of the real system. It may become necessary to use a simplified representation of the real system, which will be referred to as a *model* in the following. *Modeling* is the process of creating this desired simplified representation. Here the term model is used in the same sense as in [Shannon 1975] and [Schmidt 1985].

> **Definition**: A model is an abstract and idealized replication of a real system, which reflects all of its relevant properties with sufficient accuracy with respect to the intended purpose.

A major feature of a model is that it is "easier" to handle than the real system itself. Reasons for modeling vary; models can be constructed of any real system. The real system may be

1. completely available, examinable, analyzable, and modifiable, but it may be too expensive or time-consuming to do so;
2. observable, but not modifiable due to, e.g., a lack of technical possibilities, danger to human life and/or nature, extremely high costs, or other reasons;
3. non-existent or only partially existent; its subsystems may have properties 1. or 2. above.

The impact of available system knowledge on modeling is discussed in section 4.1. In accordance with the system theoretic approach introduced in section 1.1, during the modeling process always only a cutout of reality is considered. A limited subset of real entities is deliber-

ately chosen, and influences from others, which seem to be only loosely or not at all coupled to the chosen entities of interest are ignored. This *cutout of reality* interacts with its environment over the *system border* (environmental influences). External influences to this real system are considered to be its *input*, and the observed behavior of the parameters of interest its *output*.
During modeling, an abstract and idealized replication of the real system is created. The level of detail of the model is indicated by the degree or level of *abstraction* under which reality is perceived (*resolution*). All abstraction during examination or analysis must be conducted in a purpose-oriented manner, always giving consideration to the intended purpose of model use. If the degree of abstraction and idealization is too high, model accuracy suffers; if the degree of abstraction is too low, the model itself becomes too complex to be handled.
The real system may not only be viewed at different degrees or levels of abstraction, but also from different *perspectives*. Particular properties of the real system thereby become unimportant for the intended modeling purpose and can be ignored, which leads to further decrease of complexity. The relevant properties of the real system are identified as *attributes* of the model, submodel, or object respectively. Additionally, processes or objects can be simplified by assuming ideal conditions, which also simplify their description (*idealization*). This freedom in creating a model implies that for one real system there can be an unlimited number of different models. There is no single solution for modeling a real system.

1.2.1 Physical vs. Symbolic Models

One differentiates between *physical* (or material) *models*, which have properties similar to those of the real system due to their material or physical construction, and *symbolic models*, which provide a textual, graphical, mathematical, logical, formal or otherwise symbolic description of the structure and behavior of the real system [IEEE 610.3 1989]. This thesis is restricted to quantitative symbolic models only.

1.2.2 Symbolic Description of Structure and Behavior

A symbolic model and its submodels (implicitly or explicitly) contain descriptions of the perceived, observed, assumed, or otherwise detected structure and behavior of the real system and its subsystems. If no behavior description is available, the symbolic model is considered to be a *static model*, otherwise it is a *dynamic model*.

- The *structure description* consists of all information required to completely describe the modeled elements of the real system and their static interdependencies. This includes the definition of all objects or entities, their attributes, submodels, the organization of objects and submodels in submodels on the next higher level in the composition hierarchy (aggregation and de-aggregation, “has a”-dependencies), and the inheritance hierarchies of objects (“is a”-dependencies).
- The *behavior description* (dynamic models only) is given by the functional description of the dependencies between the input, the internal state, and the output of an object or submodel, and the interactions between the submodels on each layer of the submodel hierarchy.

Dynamic models are the main focus of this thesis, but most of the ideas presented can be transferred to static models without modification.

1.2.3 Hierarchical Decomposition

Similar to the system-of-systems decomposition of a real system introduced in section 1.1, here the model is considered as a model-of-models. Submodels can be decomposed into submodels of the next lower level, or integrated or composed to submodels of the next higher level, yielding a composition or *submodel hierarchy*. In analogy to [Zeigler, Praehofer, and

Kim 2000], submodels which are not decomposed any further (objects or entities) are called *atomic submodels* in the following. Submodel composition is finished with the *overall model* as the highest level of integration. The composition hierarchy explains how the behavior of the overall model is a result of the behavior of its submodels by integrating their behaviors.
The selection of the different levels of abstraction within a submodel hierarchy mainly depends on the input and output of interest and on knowledge about the dependencies and subsystem interactions in the real system. For example, following a top-down approach for a description of minimal complexity, if the functional dependencies between input and output can directly be formulated with sufficient accuracy, no further refinement is required – this leads to a hierarchy depth of one. Otherwise the subsystems at the next lower level of abstraction and their dependencies must be identified, and the model is composed of these less abstract submodels with their own I/O transition description, if available. The top-down decomposition process may continue until the behavior of the system can be described as the integrated behavior of its subsystems. Alternatively, the submodel hierarchy may be created from the bottom-up, starting with modeling the defined atomic subsystems and integrating them into submodels on higher hierarchical levels. However, the approach to the creation of the submodel hierarchy is only of secondary importance for the scope of this thesis; it is the *existence of the submodel hierarchy* that matters.

1.2.4 Continuous and Discrete Models

Symbolic models are classified as *continuous models* and *discrete models*. Whether a symbolic model belongs to one class or the other depends on the domains of its input, output, and state parameters and the chosen representation of time (dynamic models only). If all attributes are continuous variables, the model is considered to be a state continuous model, and, if the system behavior is modeled over continuous time, the model is considered to be time continuous. Hybrid models, which combine both continuous and discrete attributes are common. Here, both continuous and discrete models are considered.

1.2.5 Stochastic and Deterministic Models

Symbolic models may also be *deterministic* or *stochastic*. If the behavior of the model and the model output are only and unambiguously defined by the model input and the current internal state, this model is called a deterministic model. If the behavior description of the model contains random elements, the model is called a stochastic model. Hybrid models, which combine deterministic and stochastic behavior descriptions are common, and are treated as stochastic models. Here, both stochastic and deterministic models are considered.

1.2.6 Representation Forms of a Model

To deal with varying purposes of model descriptions and the necessity to communicate different types of information about the model, three different *representation forms of a model* are distinguished here: the *Conceptual Model*, the *Formal Model*, and the *Executable Model*. Their dependencies are sketched in Figure 1. (Their developmental relationship is discussed in section 1.4.6 and depicted in Figure 2.)

- The *Conceptual Model* describes the abstracted and idealized representation of the real system and holds all concepts of the model, i.e., its decomposition into interacting subsystems, the representation of properties of interest in the form of attributes, the degree of abstraction and idealization, and the rationale and reasoning that led to the chosen representation of the real system in the language of the model's application domain. The Conceptual Model serves as communication basis between those familiar with the application domain of the model and the modeler, and provides insight into the ideas behind the model, the motivation for modeling properties, e.g., chosen sys-

tem border, idealization, and abstraction. This insight is essential for comprehension and examination of the model as a representation of the real system.

- The *Formal Model* is the formalized description of the Conceptual Model, compliant with a well-defined modeling formalism, expresses the Conceptual Model quantitatively and unambiguously, and thereby prepares several methods for its solution. Because the Formal Model is a solution-oriented, implementation-independent, unmistakable description of the Conceptual Model, it provides the basis for transformation of the model into the Executable Model.
- The *Executable Model* technically implements the Formal Model and provides the additional information that allows the model to be executed and operated on a computer or in a network of computers. The additional information includes, for example, memory allocation, variable data type declaration, calls of operating system procedures, and communication protocols as typically required in development and execution environments.

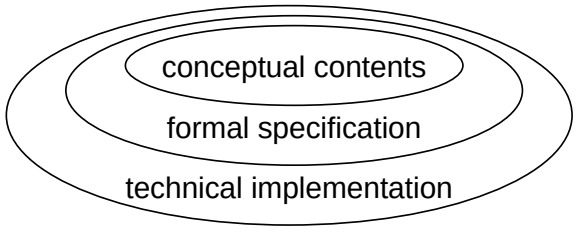


Figure 1: Embedded representation forms of a model

The stage to which a symbolic model description is developed, depends entirely on its purpose. If the only purpose of the model is to visualize qualitative structural dependencies to support understanding and managing of the real system by giving an overview, it may be sufficient to stop with the development of the Conceptual Model. For gaining any type of quantitative results (quantitatively "solving the model"), formal information, which allows the application of a well-defined (mathematical) solution technique, is added to the Conceptual Model, yielding the Formal Model.
A model may be solved by several means, including analytic methods or simulation (see also section 1.2.6). There are advantages and disadvantages to both solution approaches, as briefly discussed in the following section.

1.3 Model Solving

If a Formal Model with a mathematical foundation is available, the model may be mathematically "solved", i.e. quantitative results that characterize the behavior of the real system are calculated. Reasons for replacing the observation of the real system by a mathematical calculation of system behavior include:

- The behavior of the real system may be too fast or slow to allow observation;
- the behavior one whishes to examine would destroy or seriously damage the irreplaceable or expensive real system or its environment;
- current technology does not (yet) allow the direct observation of the behavior (the cause), but only of caused phenomena; and
- the real system does not (yet) exist.

Under certain given circumstances it is possible to solve a model analytically; however, simulation is the more flexibly applicable solution method.

1.3.1 Analytic Solution Methods

To solve a model analytically, a complete mathematical description of the model is required. Numerous modeling formalisms are available to support the creation of the mathematical description, for example Markov Chains or Queuing Nets [Hiller and Liebermann 1997]. Often based on state probabilities, characteristic values for the models can be exactly calculated or approximated without requiring large amounts of memory or computation power.
However, the applicability of analytical solution methods is limited. To allow its analytical solution, a Conceptual Model must be transformed into a Formal Model (i.e., described according to a formalism), for which well-defined solution methods are available. The limited expressiveness of the formalism and the restricted choice of stochastic input models may force the modeler to abstract and idealize in a manner that prohibits the inclusion of relevant aspects of the real system in the Formal Model [Brendel and Schäfer 2002]. In addition, even if the characteristics of the chosen input model support an analytical solution, and the Conceptual Model is properly formalized, its structure and behavior description may be too complicated for analytical solution. Although there is strong tool support for analytical solution of specific Formal Models (e.g. SHARP, [Sahner, Trivedi and Puliafito 1998]), given the requirements of industry and military discussed in section 1, the exploration of alternatives seems to be advisable. This document focuses on computer-based simulation, considering the analytical solution of a model only as a source of comparison data for the evaluation of a particular simulation approach.

1.3.2 Computer-Based Simulation

If an analytical solution of the model is not feasible due to the limitations of the modeling formalism or model complexity, or not desirable (e.g., an interactive virtual stimulus for a human or a technical device is required), one may approximate the behavior of the real system by executing the model over time, and subsequently or interactively draw conclusions about reality from the observed dynamic behavior of the model.
In the context of this thesis, *simulation* means experimentation with a model. A *simulation run* is a single execution of the model, yielding *simulation results* or model output. The execution of a *simulation experiment* consists of a set of *simulation runs*, which must be well planned, and requires an *experiment design* for the model similar to the setup of a real experiment. If the experiment design is unsuitable, results from simulation may not allow the desired increase in knowledge about the real world, or may be statistically insignificant.
There are numerous distinct simulation methods and techniques available. The selection of a particular simulation method depends on the type of the model and the intended use of the simulation results. This thesis concentrates on *computer-based simulation*. Here, simulation is defined in analogy to [IEEE 610.3 1989].

> **Definition**: Simulation is the execution of a model that behaves similar to the real system when provided a set of controlled inputs over time.

For computer-based simulation, the behavior of the symbolic model is calculated by a computer. Within this thesis, *model behavior* is the dynamic change of the model output over time, as a function of model input, and, where applicable, the internal model state or random elements.
The following taxonomy is used in this document – all of the concepts listed below are considered in the thesis:

- *Quasi-continuous* vs. *discrete simulation*: Continuous simulation requires a continuous model (with a continuous state space) and continuous advancement of time. If the model's state space is discrete, or time advances in discrete time steps, the simulation is called discrete simulation. Theoretically, on a digital computer no continuous simu-

lation is possible, but if the steps of time advancement and the discretization of the state space are sufficiently small, continuous simulation may be approximated (quasi-continuous). It is not unusual to have both quasi-continuous and discrete advancement of time in one simulation together.

- *Time-stepped* vs. *discrete event simulation*: If in a simulation time advances in time intervals ("steps") of equal length, this simulation is called time-stepped. Time-stepped simulation is scaled to wall clock (or real) time, and supports interaction between a user, physical models, or other real systems and the model (in-the-loop simulation). In discrete event simulation, simulation time "jumps" from event to event. Events are typically triggered by conditions and may be scheduled and caused from outside the (sub-) model or take place internally as a consequence of interactions among the objects or submodels. Time advancement may be both time-stepped or based on event occurrence for separate submodels contained in a model.
- *Deterministic* vs. *stochastic simulation*: This depends on whether a stochastic or deterministic model is used. Whenever a deterministic model is executed, the simulation results depend exclusively on the initial state of the model and the input data. Random decisions that must be made when executing a stochastic model rely on *random numbers* that are "drawn" during each simulation run. Depending on the random numbers, simulation results in stochastic simulation can vary with each run, even if the input data does not change. To get statistically significant results, experimentation with a stochastic model must be repeated a sufficient number of times.

In [Shannon 1975] or [Schmidt 1985] a more detailed introduction to simulation is provided.

1.4 Model Development and Execution Process

In the research field M&S, the structured planning and execution of activities that are required for model development and simulation are called the Model Development and Execution Process. M&S methodology evolved over decades, which has resulted in highly specialized model development processes for different application areas. General M&S research has also advanced, yielding different types of basic processes.

The overall purpose of a process description is to support structuring, organizing, and managing the products and activities within the process. Stages of model development are identified and associated with *intermediate or interim products* that result from each stage. The diversity of the information about the model explicitly available at these stages provides additional insight into the final product – the model and the simulation results.

To increase the credibility of models and simulation results in a most effective and efficient manner, quality assurance measures and V&V activities should be embedded into the model development process, according to the discussion in chapter 3. As a foundation for the V&V framework that is developed in this document, here several processes of model development and execution are briefly reviewed for their potentials and limitations in the following. Then, based on this review, the selection of the process model for the creation of a V&V framework is motivated.

1.4.1 Modeling Activities and Intermediate Products

The activities conducted during model development can be organized in phases, according to their contents and chronological occurrence. In the scientific literature numerous model development processes are presented, which most often can be distinguished by the selection of phases and the way in which the feedback between the phases is organized. Not only the iteration of development phases, but also the resolution or level of detail of the activity descriptions making up each phase can vary. This mainly depends on the motivation of the creator of

the model development process (which again is a purpose-oriented model) and the aspects of model development that seem most important to this particular person or group.
Generally speaking for M&S, to yield meaningful simulation results it is essential that:

- the purpose of modeling and the application of simulation results is communicated correctly and completely, and is documented precisely,
- the real system subjected to modeling is analyzed (including data measurement), abstracted, and idealized giving consideration to the intended purpose with well-defined requirements, and is documented with all its constraints and limitations formulated explicitly,
- the Conceptual Model of the real system is specified completely and unambiguously using formal and/or mathematical methods as a Formal Model,
- the Executable Model that will actually substitute the real system for experimentation is free of implementation errors and correctly implements the Formal Model, and
- the experimental application of the Executable Model is conducted with suitable input data, within the model's limitations in well-designed experiments.

The analysis of the real system and finally the creation of the model are time and cost intensive, especially for large-scale systems. To avoid mistakes during the development of the model, the development process must be manageable, which requires clear structuring of all developmental activities. In the area of software engineering – years ago – this realization resulted in actions: Significant standards or quasi-standards like the ISO9000 [Thaller 2000], discussed in section 3.2.2, or the Capability Maturity Model [Paulk et. al. 1993], discussed in section 3.2.1, require clear and unambiguous specification and documentation of the processes, which an organization should follow for the development of a software product.
The different activities executed during the stepwise development of the model are summarized according to their aim and chronological order, therefore allowing the decomposition of model development into phases. In the literature [Thaller 2000; Scrudder et. al. 1998; Lehmann et. al. 2002; Bundesministerium für Verteidigung 1997; Balci 1997b; Shannon 1975; Schmidt 1985], numerous representations of model development processes exist, which can be distinguished by the way in which activities are mapped to phases, by the knowledge required to perform those activities, the types of feedback loops reflecting the possible repetition of particular phases (iteration), and by the contents of the intermediate products constituting the result of the developmental work of each phase. The resolution of the phase-wise decomposition of model development varies depending on the perceived relative importance of particular activities.
In the literature there are several classes of process models for the development of models, software products, or simulation results. They include

- *Waterfall processes*, as discussed in section 1.4.2,
- *V-type processes*, as discussed in section 1.4.3,
- *Life cycle processes*, as discussed in section 1.4.4, and
- *Spiral processes*, as discussed in section 1.4.5.

As the model development process itself is just a model of model development, there is no universally accepted presentation of the model development process. When choosing a model development process, one is responsible for choosing a process most suitable for the intended purpose. To motivate the selection of a process model in the context of this thesis, the following four different classes of process models are presented, which are used or may be used in modified form for the description of model development. At the end of this section the model development and simulation process is introduced that will be used for the remainder of the thesis.

1.4.2 "Waterfall" Processes

Waterfall processes are linear representations of model development with a very limited number of feedback loops. A sample waterfall process for software development is provided in [Boehm 1981], samples for M&S waterfall processes in [Shannon 1975] and [IEEE 1516 2001].

Model development processes of the waterfall type are clear, comprehensive, and easy to handle. The unambiguous separation between the phases theoretically allows a clear assignment of activities to phases. However, in reality such an idealized flow in model development is rare, which leads to the conclusion that the waterfall model is not suitable to describe the real process of model development *as it is*. This type of an idealized description should be considered more as a desirable goal, then as valid description of today's practice. For conceptual model-theoretic considerations, this type of model development process seems to be suitable.

1.4.3 V-Type Processes

Model development processes in the "V-Style" focus on direct mirroring of its early and late phases. In addition to the direct steps back to the predecessor phase, special feedback loops to associated earlier phases are included. A sample of a V-type IT system development process is provided in [Bundesministerium für Verteidigung 1997]. The phases definition of the submodule SE of the "V-Modell", the product flow and especially the carefully chosen feedback loops qualify it as a practically applicable guideline for the management of system development activities.

The clear separation of the development phases leads to a high degree of clarity of model development processes in "V"-Form, too. Again, in theory an unambiguous assignment of activities to each phase is possible. Giving consideration to QA measures, the V form shows the dependencies of the contents of the phases. However, even if feedback loops seem to be numerous, a significant number of additionally possible feedback loops is (deliberately) ignored.

1.4.4 Life Cycle Processes

In many cases the execution of a model and the generation of simulation results do not terminate a "model's life". After the evaluation and interpretation of simulation results and the formulation and implementation of conclusions for reality, the requirements for the model may be slightly modified and the model accordingly adapted, to support new examinations by simulation. Examples of M&S live cycle processes are provided in [Balci 1997b; Sargent 1999; Robinson 1999].

In comparison to waterfall processes or V-type processes, life cycles usually are less clearly structured, although the precise separation of the phases supports clarity. A model life cycle is especially useful, if one needs to include the dependencies between the modeling process, the thereby gained new system knowledge, and the consequences for the real system. Model life cycles describe the process of the permanent modification of a model after its first use, which supports the representation of configuration control.

1.4.5 Spiral Processes

The clear separation between phases of model development is abandoned with the spiral process. Consequently the creation of intermediate products associated with the end of a phase is not strictly demanded, nor are intermediate products defined. Spiral models support the visualization of iteration cycles and the chronological order of modeling activities. Without phases and with iteration cycles there are no feedback loops required.

The phase-less incremental spiral model is a very realistic representation of the model development process as it currently seems to be conducted in most enterprises in the USA and

Europe [Thaller 2000]. Unfortunately, the lack of clarity and of mandatory intermediate products disqualify this process model for conceptual and theoretical considerations with respect to QA and integration of V&V activities. For example, [Lee and O'Keefe 1994] use a spiral process model for the discussion of V&V activities during the development of expert systems, but when addressing developmental activities, they implicitly identify "sectors" of the spiral and thereby reduce the spiral process model to a life cycle process model.

1.4.6 The Chosen Model Development and Execution Process

As an indispensable prerequisite for a V&V process that leads to a systematic increase in credibility of the model or the model results, there needs to be the possibility of precisely defining intermediate products and the activities performed on these products (see section 3.1.1). Therefore, for M&S development and use the author assumes a clearly structured model development and execution process, which satisfies the minimum requirement of providing well-defined intermediate products. To avoid unnecessary complications, the *waterfall process* type is chosen. It is the least complicated of the reviewed process types, which is *organized in phases* with *one intermediate product (IP) associated with each phase*, as depicted in Figure 2. Each intermediate product documents all results of the associated phase and serves as an input to the next phase. Iterations due to the detection of inadequacy or incorrectness will be dealt in the V&V process. Iterations due to changes in requirements are deliberately ignored, as they are not an issue of V&V, but of configuration management, which is out of the scope of this document.

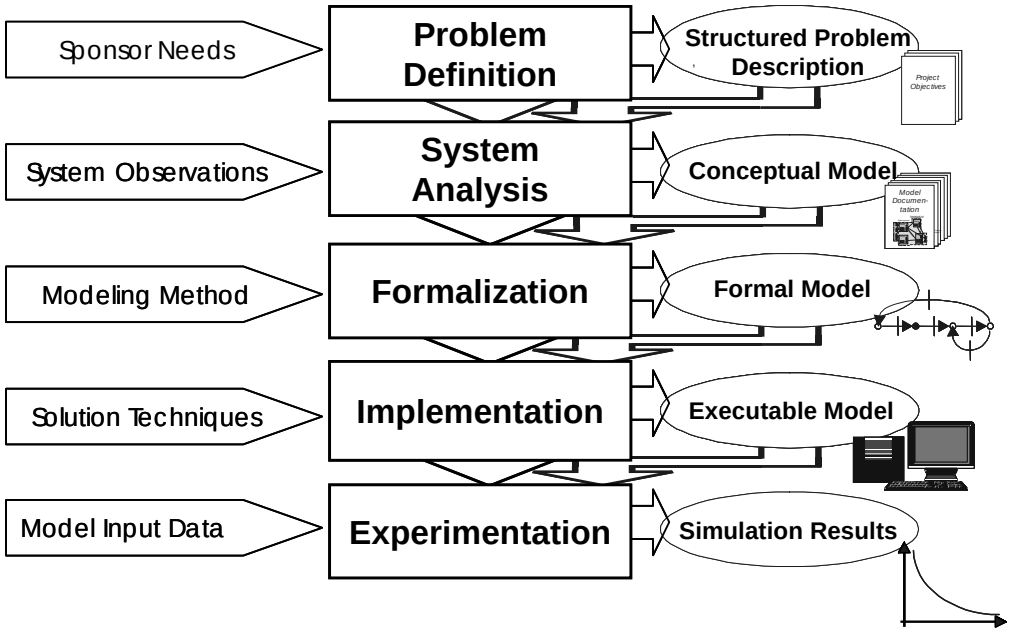


Figure 2: The chosen model development and execution process

The process depicted in Figure 2 is organized as follows:

- As a starting point, the reasons why the model is to be built to substitute a real system (for some analysis, operation, training, or experimentation) and direct goals that shall be achieved by M&S should be documented in an initial product, which is called the *Sponsor Needs* (SN). Usually the Sponsor Needs are expressed informally and in language the sponsor is familiar with.
- Then, during the *problem definition phase*, required interfaces and contents of the model have to be defined and documented as a *Structured Problem Description* (SPD).

- As a result of the *system analysis phase*, the perceived, observed, or assumed structure and behavior of the real system, including all of its relevant subsystems, which have to be reflected by the model, are documented as the *Conceptual Model* (CM). It expresses and communicates the "idea behind the model" in language the domain experts are familiar with.
- Subsequently, to formulate the concept of the model precisely and unambiguously, and to lay a foundation for the quantitative solution of the model, mathematical or formal modeling methods are used. The result of the *formalization phase* is documented as the *Formal Model* (FM).
- Finally, after the selection of a particular solution technique, the *implementation phase* yields an *Executable Model* (EM). For the remainder of this document, the solution technique is assumed to be a simulation technique, as introduced in section 1.3.2. The Executable Model is executed by a digital machine and reflects both conceptual and mathematical solution, while dealing with a wide range of technical challenges.
- After simulation (*experimentation phase* or model operation) the model output (model results, or when simulating, *Simulation Results* (SR)) may be used to substitute or supplement observations or measurements of the real system, if the simulation credibly represents the behavior of the real system.

Each of the developmental steps contains new important aspects of model development or use, which are well hidden in the final product. If the results of the phases are documented by the intermediate products, the final model can be examined from different perspectives by analyzing the intermediate products. The detailed specification of the intermediate products *Structured Problem Description*, *Conceptual Model*, *Formal Model*, *Executable Model*, and *Simulation Results* developed during this thesis can be found in Appendix A.
The author is aware of the fact that this model development and execution process is a heavily idealized description of this process in every day practice. But it serves as solid foundation for the integration of V&V activities, and is sufficient to populate the three-dimensional model information space introduced in section 4.3.

1.4.7 Parties Involved

Because the development of a model and the execution of simulation experiments require numerous activities, quality assurance of models and simulations becomes a non-trivial field requiring the skill and knowledge from several areas and usually involves various individuals. However, as an extreme, one single person could be responsible for the completion of all tasks during model development, but in other cases teams develop the model. Because the assignment of tasks to persons or individuals should be up to the developer, roles and associated responsibilities within the model development process are identified. A role is outlined by

- the knowledge and skill required to complete the associated tasks, and
- the responsibility taken in the model development and execution process.

The role model does not determine, whether one person plays several roles, or several persons share one role. However, particular roles require a sufficient distance between the individuals or teams performing them. Roles are defined in a similar form in [Defense Modeling and Simulation Office 1996 and 2000] and [Bundesministerium für Verteidigung 1997]. This document addresses the responsibilities of the following actors:

- User / Operator: The model user or operator conducts the experiments with the model and is responsible for precisely specifying its required functionality. The user applies the model to generate simulation results. Typically there is a close relationship between the user and the sponsor who provides the funding for development.

- Sponsor / Beneficiary: The sponsor initiates the modeling process with the intention to solve a problem in his application domain by extending his knowledge base with simulation results. He provides funding for M&S development and is responsible for the precise description of the intended purpose of the model or simulation. (Financial aspects are out of scope of the thesis and are not further discussed, thus only the beneficiary aspect of the role is of importance.)
- Simulation Project Manager: This role is responsible for the administrative aspects of the development of a suitable and correct model or simulation results. The actor manages the overall model development process, ensures that the chosen model development process is correctly utilized, and ensures that all intermediate products are available at the project milestones.
- System Analyst or Subject Matter Expert (SME): This role contributes to the modeling process by adding knowledge about the application domain and is responsible for the analysis of the real system. The actor is an expert on a cutout of the application domain and knows how to set up tests for the real system.
- Modeling Expert: This role knows theory and practice of M&S, modeling paradigms, solution techniques, and tools. The actor is responsible for the creation of the Conceptual and the Formal Model, i.e., mainly responsible for system analysis and model formalization.
- Programming Expert: This role is capable of encoding the simulation model. The actor knows the execution platforms, operating systems, and development environments, and is mainly responsible for the implementation.

It is the responsibility of these roles to create a model and simulation results in close cooperation, in a manner such that any pitfalls are avoided.

1.5 Verification and Validation of Models and Simulation Results

The purpose of M&S is to represent a real system in order to draw conclusions about the real system by experimentation with a model. Thus, the direct correlation between the model, an intended purpose of model use, and a clearly identified real system are among the key characteristics of simulation. The term “simulation” implies a claim to represent the behavior of a real system as it is or as it could be. (The direct association to a real system distinguishes computer-based simulation from, e.g., computer games.) As decisions that heavily impact the real world rely increasingly on models or simulation results, the more important their *correctness* (section 1.5.1) and *suitability* (section 1.5.2) becomes. Suitability refers to the concepts of *capability*, *fidelity*, and *accuracy*, while correctness refers to *consistency* and *completeness* (see Figure 3).

The growing role of M&S implies that measures must be taken to ensure the correctness and suitability of models and simulation results. As neither suitability nor correctness can be proven in most cases, the *credibility* of a model or simulation results is of major importance. The credibility of a model is based on the *perceived suitability* and the *perceived correctness* of all intermediate products created during model development. The correctness and suitability of simulation results require correctness and suitability of the model and its embedded data, but also suitable and correct runtime input data and use or operation of the model. *Verification* and *validation* aim to increase the credibility of models and simulation results by providing *evidence* and *indication* of correctness and suitability. The dependencies between the terms introduced above are illustrated in Figure 3.

1.5.1 Correctness and Verification

When using a model or simulation results, one expects them to be internally consistent (self-consistent), correctly described in all their different representation forms, and completely consistent with each other. Through verification of a model, its input and its embedded data, and its simulation results one determines, whether any transformation or description was performed according to well-defined rules and specifications. Verification deals with the examination of *correctness* (including consistency and completeness) with respect to these well-defined rules. It addresses the question, whether something "has been built right" [Balci 1990; Shannon 1975]. Verification always requires some kind of documented specification or description, which is transformed into another specification or description, and unambiguously (ideally formally) defined rules.

The term "correctness" is also used in related fields, e.g., in software verification. In this case a code block is considered to be partially correct, if under the assumption that a formally specified pre-condition holds, the execution of the code block results in the satisfaction of a formally specified post-condition [Berghammer 2001]. If the execution of the code block always terminates, too, it is even considered to be totally correct. However, software development is only one aspect of M&S. Especially during the early phases of the model development process (section 1.4.6), information and knowledge is documented that successfully resists formalization using a syntactically and semantically specified language of limited expressiveness. Thus, in the context of verification of models and simulation results, the concept of correctness is used in a wider sense than in software verification.

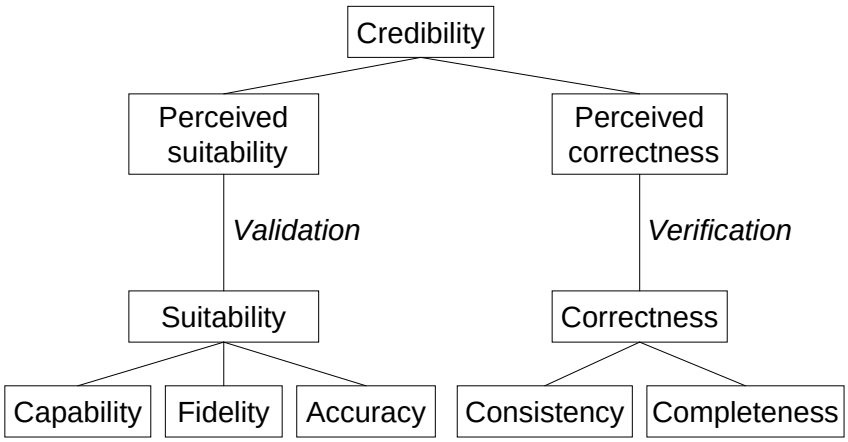


Figure 3: Dependencies between V&V related terms

[Balci 1998a] defines model verification as "substantiating that the model is transformed from one form into another, as intended, with sufficient accuracy", and thereby stresses the consistency requirement among the intermediate products. [Zeigler, Praehofer, and Kim 2000] keep distance from the definition of verification, but treat this problem under simulator correctness: "A simulator correctly simulates a model, if it is guaranteed to faithfully generate the model's output trajectory given its initial state and its input trajectory." They stress the behavioral (actually model execution) aspect and thereby approaches verification from another direction. As there is no commonly accepted definition for model verification in M&S literature, the use of the term "verification" in the context of this thesis will be as follows:

> **Definition**: Model verification is the process of demonstrating that a model is correctly represented and was transformed correctly from one representation form into another, according to all transformation and representation rules, requirements, and constraints.

Considering the representation forms of a model introduced in section 1.2.6, model verification includes ensuring that

- the Structured Problem Description is met by the Conceptual Model,
- the Conceptual Model is completely formalized as Formal Model and consistent with the chosen modeling formalism, and
- the Formal Model was implemented correctly and completely.

The main difference from validation (see section 1.5.2) is that the references to the real world and the intended purpose are not subject of verification. Precisely formulated specifications of the requirements, of the model in all its representation forms, and of the rule set are examined exclusively.
Theoretically it is possible to verify a Formal Model completely, but in current practice imprecise specification, model complexity, or lack of knowledge about the model (worst case: black box) preclude exhaustive verification. This situation requires a positive approach to verification. When doing non-exhaustive verification, the verification aims should be chosen in a way that allows the justified conclusion that the correctness of the examined parts of the model indicates the correctness of the overall model. Please note the difference from *model falsification*: While for falsification it is sufficient to identify a single counterexample, for exhaustive verification it must be proven that there is no counterexample. As long as this proof is not feasible, the non-existence of a counterexample can only be made plausible. This problem is again addressed in section 4.4.
If one is going to use simulation results, verifying the model itself is necessary, but not sufficient. [Pawlikowski, Jeong, and Lee 2002] point out that in the field of telecommunication the use or operation of models and the evaluation of simulation results occur in an unacceptable manner. It must be assumed that this example can also be transferred to other similar application domains. Thus, to get reliable quantitative results from the model, *verification of input (run time) data and embedded data* and *verification of the experiment* are also required. This includes the demonstration that the input and embedded data cover the required value domain, have sufficient accuracy (decimal places, measurement accuracy, measurement error), dimension, and unit. If aggregated data is used, it must be demonstrated that aggregation was conducted correctly and according to the aggregation strategy. It should be ensured that the data is organized in compliance with the required data models and in the correct technical format.
There are numerous sources for methods and techniques for the verification of models and simulation results, e.g., [Defense Modeling and Simulation Office 1996 and 2000; Katoen 1999; Balci 1990; Liggesmeyer, Sneed, and Spillner 1992; Myers 1979]. These techniques are usually highly specialized, as checking for (absolute) correctness requires an unambiguous description of the item subjected to examination and the specification of rules. This strongly encourages tool support – however, the high degree of specialization implies only limited opportunities for application of such tools, and the range of detectable errors is small. The exploration of these limits, the identification of the benefits, and the ability to create evidence that inspires belief in the correctness of the model or the simulation results are among the goals of this thesis.

1.5.2 Suitability and Validation

When using a model or simulation results, one expects them to be suitable for the intended purpose, within acceptable constraints. Validation of models or simulation results is performed with respect to a real system, and always with respect to the intended purpose of model use. Validation deals with the suitability of the model (including capability, fidelity, and accuracy) and addresses the question of “whether the right model has been built” [Balci 1990; Shannon 1975]. The main difference from verification is that the model is checked for its suitability as a substitute for the real system with respect to its intended use (assuming its

correctness) – rules and precise specifications play a subordinate role. This is due to the problem that it is difficult, if not impossible, to precisely specify vague terms like "intended purpose", "suitability", or "acceptable difference to reality", and therefore to generate precise rules for determination of validity.
In the M&S literature there is no common understanding of the term "validation". [Shannon 1975] defines validation as "the process of bringing to an acceptable level the user's confidence that any inference about a system derived from the simulation is correct", and avoids to address any property of a model (I/O, structure, or behavior). [Balci 1998a] writes that "model validation is substantiating that within its domain of applicability the model behaves with satisfactory accuracy consistent with the study objectives" and concentrates on behavior accuracy, which is an extremely small subset of Shannon's definition. [Zeigler, Praehofer, and Kim 2000] explains it as "the concept of validity answers the question of whether it is impossible to distinguish the model and system in the experimental frame of interest".
As there is no commonly accepted definition for model validation, the use of the term "validation" in the context of this thesis will be as follows:

> **Definition**: Model validation is the process of demonstrating that a model and its behavior are suitable representations of the real system and its behavior with respect to an intended purpose of model application.

Reality in all of its complexity cannot be expressed; it is represented by theoretic assumptions and (mental) models of reality, or by (empirically) measured data of related systems, real subsystems, or the real systems, which serve as *referent*. Thus validation of the representation of structure and behavior of the real system provided by the model usually is limited to *the failed attempt at its systematic falsification.* This situation requires a positive approach to validation. When doing validation, the validation aims should be chosen in a way that allows the justifiable conclusion that the suitability of the examined parts of the model indicates the suitability of the overall model. This thesis contributes to this issue.
In order to generate output, a model requires input data and experimentation; one should also take care to be sure that the data used and the process of conducting experiments are suitable. This includes the demonstration that the input data (stream) suitably substitutes for the influences generated by the environment on the modeled properties and that embedded data suitably describes quantitative properties of the real system. When input data is created from raw data by aggregation, it must be ensured that any loss of information does not invalidate the data. [Sargent 1999] points out the importance of valid data for conceptual modeling. Experiment validation also includes demonstrating that the experiment setups are representative of real situations. If the conditions under which the simulation is run are unrealistic, simulation results do not allow conclusions about reality to be drawn, no matter how suitable and correct model and data are.
There are numerous techniques for the validation of models and simulations [Defense Modeling and Simulation Office 1996 and 2000; Schnepf 1990]. However, their specification often is vague, and their field of application wide, leaving a great amount of subjectivity in the judgment of suitability. To point out the benefits and limitations of their application, and their ability to create indications or evidence that increase the credibility of the model or simulation results is among the goals of this thesis.

1.5.3 Demarcation: Software Quality Assurance

As stated in section 1.3.2, in the context of this thesis computer-based simulation is the solution approach for the generation of quantitative model results. This implies that the model is implemented in software to allow its execution on a computer (Executable Model – see also section 1.4.6 and Appendix A). However, model development does not equal software devel-

opment; software development is one part of model development. For the remainder of the document it is distinguished between software quality assurance (software QA) and V&V of models and simulation results. Software QA measures need to be taken to ensure that the software is stable, correctly accesses peripheral devices, does not hang up or crash, does not crash the computer system, interacts correctly with other software components of the operating system or other applications, terminates, accepts operator input, or in other words, that the software "runs". Software QA procedures and measures can be found in the appropriate literature, e.g., [Myers 1979; Liggesmeyer, Sneed, and Spillner 1992; Beizer 1990; DeMillo et al. 1987], and software QA is out of the scope of this document. For the V&V of the Executable Model, the existence of a "running" Executable Models that allow the observation of model behavior is assumed. Methods for M&S V&V are likely to reveal software bugs that slipped through software QA as a side benefit, and thereby are complementary to software QA, but their efficient application requires software of a sufficiently high quality, i.e., "running" software.

1.5.4 Credibility

The nature of validation and the technical constraints on verification imply that currently one cannot declare a model or simulation to *be* absolutely correct and suitable; one can only express the *belief that the model is correct and suitable,* or develop an expression of the *credibility* of the model from one's point of view.
As there is no proof of correctness and suitability with non-exhaustive V&V, one may want to determine the *degree of confidence* in the model. This confidence intuitively is based on the probability that an existing incorrectness or unsuitability is not detected during V&V. This probability again depends on the *intensity* and *rigor* of the V&V applied, which is indicated by the V&V output products (evidence) created. In the context of this document, rigor provides a qualitative measure of the variety of model contents examined (*breadth*), and intensity qualitatively measures the thoroughness of the examination of a particular model content (*depth*). This will again be referred to in section 4.1.2

> **Definition**: The credibility of a model or simulation results is an expression of the degree to which one is convinced that a particular model or particular set of simulation results are suitable for an intended purpose and correct.

Conducting V&V to establish the credibility of a model or simulation results is more than simply increasing model quality. The user must recognize the effort spent on and the results gained by V&V. *Doing* V&V is necessary, but not sufficient. The V&V efforts have to be *communicated* to the user, who in turn needs to be convinced of the correctness and suitability of the model or simulation results. This requires careful documentation and configuration control of every little piece of V&V conducted.
The introduction of a vague term like credibility is intended to ensure that the (hopefully small) amount of subjectivity in the decision of accepting a model or simulation results for an intended purpose is never forgotten. An in-depth discussion about credibility can be found in section 4.4.

1.5.5 Accreditation

In today's M&S community, the terms "verification" and "validation" are usually mentioned as part of the triplet "VV&A", with the "A" standing for "accreditation". Accreditation is a bureaucratic act, during which a model or model results officially are declared as acceptable for a specific intended purpose [Defense Modeling and Simulation Office 1996 and 2000]. Accreditation should be exclusively based on the credibility of the model. Observation of cur-

rent accreditation practice creates the impression that it can also be influenced by political interests, which should never be the case.

> **Definition**: Accreditation is the official certification that a model, simulation, or federation of models and simulations is acceptable for use for a specific purpose. [Department of Defense 1996].

An informed accreditation decision requires information about the model or simulation results, which serves as indication and evidence for their correctness and suitability; such information is generated during a systematic execution of V&V activities.
Model accreditation should be undertaken with great care; using a model for the generation of simulation results usually is not the end of its application procedure, but the results are subsequently interpreted. For proper interpretation, often deep insight into the model is required. This insight can never be replaced by an official statement that the model is “fit for purpose”. [Davis 1992] summarizes and discusses the problems that are likely to come with accreditation, including the danger that the delay caused by a bureaucratic accreditation procedure might encourage the re-use of already accredited models, which actually are *not* suitable for the intended purpose. As accreditation is a management decision, it is out of the scope of this thesis.

2 GOALS OF THIS THESIS

Modeling and Simulation possess the potential to significantly enhance traditional understanding, experimentation, examination, analysis, testing, and exploration of real systems. Although models and simulation results are widely applied, the potential for their use has not yet been exploited, because the applicability of models and simulation results – and their acceptable indirect influence on the real world – is limited by questions about their credibility. The high-level goals of current V&V research can be summarized as:

(1) Demonstrate the correctness and suitability of models and simulation results by careful and structured documentation of their systematic examination, to increase their credibility and to reduce the probability of unjustified rejection ([Balci 1998a]: Type I error).
(2) Ensure the correctness and suitability of models and simulation results by their systematic examination, to avoid their acceptance and use in case of incorrectness or unsuitability ([Balci 1998a]: Type II error).
(3) Increase the quality, and thereby the credibility of models and simulation results in general, in order make M&S even more useful then they are today.

This chapter is organized as follows: First, the most common current problems in the M&S community are summarized and general constraints of M&S V&V identified. Then the goals of this thesis are derived.

2.1 Current Deficiencies in V&V and Challenges in the Problem Field

At least in the military simulation community, currently there is no commonly accepted procedure for development, documentation, and V&V of models used for computer-based simulation. This is due to the following problems and misperceptions that have been identified through the author's own experience and in the literature [Kilikauskas 2001]. Although little explicit confirmation was found, these seem to be similar to the problems in other application domains [Pawlikowski, Jeong, and Lee 2002]. The current situation in M&S V&V can be summarized as:

- Model development is often only seen as software development;
- Model documentation is of secondary importance;
- The dependencies between the risk of model use and the required effort for risk reduction by V&V are unclear;
- The beneficial influence of constructive V&V activities on model development is underestimated;
- There is no in-depth guidance on V&V, as most available guidelines are abstract or incomplete;
- Missing documentation of conducted V&V activities precludes others from reproducing, copying, evaluating, judging, or assessing them afterwards.

Each of these deficiencies is a serious obstacle to the application of models or simulation results to real world problems, but once identified, they can be addressed. To solve these problems, managerial, organizational, psychological, and technical obstacles must be surmounted. This document is intended to contribute to the solution of the technical aspects of the problem.

2.1.1 Model Development Considered as Software Development

Encoding a model in a form that can be executed on a computer is an essential and necessary part of the development of a computer-based simulation, but not the only task. Abstracting and idealizing the real system in a problem-oriented manner is an extremely important step of model development, a fact, which is completely ignored when reducing model development to software development. Model development processes published in scientific literature [Shannon 1979; Balci 1990; Schmidt 1985] graphically visualize the main phases of model development, always including some type of problem definition and system analysis, which is then followed by model implementation and finally experimentation with the model (simulation; see also Figure 2). When clearly separating software development tasks from non-software development tasks, it becomes obvious that ideally the actual process of (symbolic) modeling is already completed when software development begins. This also implies that measures for quality assurance of software do not automatically ensure the correctness and suitability of the symbolic model, which is subsequently encoded in software (see also section 1.5.3). Therefore it is not sufficient to require a software developer to be compliant with software quality standards to increase the credibility of models and simulation results. V&V must be integrated into the complete model development and execution process.
The challenge associated with this deficiency is of educational nature mainly. It needs to be ensured that those who develop models know about their special responsibilities and the special activities and tasks that distinguish model development and use from software development.

2.1.2 Missing Model Documentation

Models that have been developed in the past, in many cases prove to be poorly documented [Gutte et. al. 2001; Könke et. al. 2000; Kilikauskas et al. 2002; Scholten 1998]. This is due to several reasons, among them

- protection of intellectual property of the developer;
- lack of explicit documentation requirements, little or no funding for model documentation, and a general ignorance of the importance of model documentation;
- missing discipline in model development.

With no or only little documentation available, the reasons that lead to the idealization and abstraction conducted during the modeling process are rarely reproducible, even if expensive re-engineering reveals the concepts of the model well hidden in the code. The process of gaining insight into the model that is required to assess the model for, e.g., re-use or model modification is costly after development was finished. Approximately one and a half decades ago, in Software Engineering the situation was similar [Smith and Wood 1987], but has changed with the introduction and spreading of several concepts, including:

- Software engineering processes, which stress the importance of requirements specification and design;
- the availability of expressive formal software specification methods as, e.g., Z [Woodcock and Davies 1996] or UML [Object Management Group 1999; Fowler 1997], which perform well as useful software specification languages;
- Quality assurance standards as the ISO 9000 family [Thaller 2000];

- Process models for software design as the V-Model [Bundesministerium für Verteidigung 1997] or the Rational Unified Process [Kruchten 2000].

Without sufficient model documentation, the process of demonstrating correctness and suitability (and thereby establishing of credibility) is expensive and does not promise to be successful. This problem is again referred to in section 4.3.
Associated with this deficiency are two major challenges. First, the sponsor's awareness of the importance of model documentation and the willingness to actually spend money on documentation needs to be increased, which can be achieved by highlighting current problems with poorly documented legacy models. Second, there also needs to be an obvious benefit for the model developer in documenting the product precisely, such as a measurable gain in productivity or increased configuration control efficiency (return of investment).

2.1.3 Under-Appreciation of Constructive Aspects of V&V

V&V do not add any new functions or features to an Executable Model, and therefore do not make the product appear more attractive at first glance. To numerous model developers, V&V is considered to be a kind of necessary evil that is conducted in the end of model development, because it is the sponsor's desire. In the worst case, the sponsor is not seriously interested in V&V either, but only follows a dictated policy. With soft milestones during the project and a hard deadline in the end, the concluding V&V activities are most likely to be reduced or even eliminated, often with the agreement of the sponsor. The constructive aspects of V&V conducted in parallel to model development are underestimated, and therefore no serious attempts are made to establish V&V activities as an integral part of the model development process.
With respect to this deficiency, the challenge lies in uncovering the constructive aspects of V&V. Useful model development processes with appropriate tool support, which integrate V&V in an efficiency increasing manner are most likely to convince model developers of the beneficial influences of V&V. The potentially positive impact of V&V on model development is again referred to in chapter 4.

2.1.4 Missing V&V Guidance and Lack of Comparability

Currently detailed and consistent guidance through the V&V process is missing, which results in a misperception of the practicality of V&V and the fear of getting lost within the V&V activities without any meaningful result. With respect to software testing that suffered from similar misperceptions, the statement "if you thought designing and coding that program was hard, you ain't seen nothing yet" was among [Myers 1979] motivations to organize software testing methods in a structured form. Although there are numerous V&V techniques from different research areas summarized in, e.g., [Defense Modeling and Simulation Office 1996 and 2000; Balci 1998a], there is a lack of awareness within the M&S community of these techniques and the necessity of tool support [Pace 1999b]. Comparison between V&V activities is difficult, and available tools are rare. This problem is again referred to in chapter 4, and the challenge associated with this deficiency is easily phrased: Provide V&V guidance.

2.1.5 Missing Estimate of Required V&V

According to [Brain 2001], the determination of the required V&V effort can be expressed as "balancing the cost of knowing against the risk of assuming". Several approaches exist to describe the dependencies between the risk of using a model or particular simulation results and the required rigor and intensity of V&V (e.g., [Mugridge 1999; Muessing, Laack, and Wrobleski 1997]). However, commonly accepted significant quantitative (and even qualitative) measures for both the intensity of V&V and the degree of risk are missing, and none of

the known quantitative approaches is based on an empirical foundation. As a consequence, a relatively precise estimation of the required V&V effort is currently not possible.
The challenge lies in the identification, formulation, and evaluation of significant quantitative measures for the risk of model use and intensity and rigor of risk-reducing V&V activities. This problem is again referred to in section 4.1.

2.1.6 Missing Documentation of V&V Activities Previously Conducted

Whenever a model developer is interested in a suitable and correct model, V&V techniques are applied in the end of the model development process, as already documented in [Shannon 1975]. Especially output validation is commonly used for the "calibration" of the model, and for the detection of errors or inadequacies in the (executable) model. However, most often the developer is satisfied with successfully detecting and eliminating detected errors and does not document the validation effort itself. In this case the V&V effort cannot be credibly communicated, and a judgment or assessment of the correctness and suitability of the model is hardly possible.
Here the challenge is to convince the sponsor of the necessity to task the developer with the documentation of the V&V plan and the conducted V&V activities, prior to the attempt to assess the credibility of a model, and to finance this additional effort. This problem is again referred to in chapter 4.

2.2 Focus of the Thesis

This thesis aims to provide an approach to systematically analyze models and simulation results for correctness and suitability, to counter the above deficiencies, and to create a foundation for informed assessment of their credibility. Based on an extensive analysis of existing approaches and standards from related areas, this thesis motivates, develops, specifies, and evaluates a framework for examining and assessing the correctness and suitability of models and simulation results, and thereby points out the similarities and dependencies between most approaches to V&V known today. To facilitate this, this thesis proposes

- a **general methodology of model development and integrated V&V**, which is examined independently from an application domain,
- **documentation requirements of intermediate products** generated during model development and framed as products of a straight-forward model development process,
- guidance through the process of V&V by a **precisely defined process**, and
- the **selection, application, and documentation of so called "V&V techniques"** and their results supported in such a manner that the drawbacks and limitations of a particular technique are mitigated by the advantages of others, and synergetic effects between the techniques are achieved.

Taking these problems and constraints into consideration, this thesis focuses on supporting the creation of meaningful V&V results, which serve as evidence or indication of model credibility, following a structured approach. To achieve this, the document is organized as follows:

- In chapter 3, the state of the art in M&S V&V is summarized, and principles of V&V are identified and discussed. The fundamental concepts of V&V developed so far are extracted from numerous pieces of related work and analyzed for their suitability for M&S V&V.
- In chapter 4, foundations for the generalized V&V process are laid by the definitions of all inputs to the process and all desired outcomes, including discussions on application risk and required V&V effort, real system knowledge, model knowledge, and the need for evidence and indications of credibility. The overall V&V aim to demonstrate

correctness and suitability of a model and its simulation results is decomposed into sets of V&V objectives. In the end of this section, the generalized V&V process is specified.

- In chapter 5, the V&V process is evaluated by its application to a sample case. For the purpose of evaluating the concepts presented in this paper a small, but non-trivial road traffic simulation model was created and implemented (Appendix C). Its documentation and the documentation of V&V conducted on the model is found in the appendices of this document.
- Chapter 6 finally concludes the document with a summary of the contents presented and the outlook for potential ongoing work.

Several products created during the work underlying this thesis are documented in the Appendices:

- Appendix A documents the V&V Triangle specification. It contains the process organization, the precise specification of the intermediate products, the identification of their potential errors, and strategies for uncovering them. V&V techniques well known from other application domains as, e.g., Software Engineering are analyzed and associated with the identified V&V objectives to support their achievement.
- Appendix B gives an overview of the identified V&V objectives and the V&V evidence made available during the V&V process.
- Appendix C documents the sample M&S project which was used as a test case for the evaluation of the V&V process.

3 THE STATE OF THE ART

This section describes the commonly accepted state of the art in V&V of models and simulation results. Although there are not many constants in M&S V&V so far, the community agrees on a set of V&V principles that have been published by several authors. Subsequently, known approaches to V&V and standards and guidelines for quality assurance in related areas are subjected to examination. The section is concluded with the identification of the basic concepts that can be extracted from the related work as a foundation for the approach taken in chapter 4.

3.1 Principles of V&V Planning and Implementation

Principles that are known throughout the V&V community [Defense Modeling and Simulation Office 1996 and 2000; Balci 1997a; Balci 1998b; Pace 1999a] serve as the foundation for the work documented in this thesis.

3.1.1 V&V as Integral Part of Model Development

If one wants to develop a suitable and correct model efficiently, V&V must be an integral part of the development process. V&V in parallel to model development supports the identification of unsuitability and incorrectness already in the early stages of model development, resulting in direct constructive feedback. Correcting errors soon after they occurred is much less expensive then allowing the error to propagate through the development process [Defense Modeling and Simulation Office 2000]. Both, the detection of errors and/or inadequacies within the model becomes more difficult the longer the model development has been completed. In addition, currently documentation available after completion of model development is incomplete and imprecise, and therefore is not an acceptable basis for “post development V&V”, making direct feedback with the person(s) in charge of the product necessary.

3.1.2 Precise Specification of the Intended Purpose

M&S are always purpose-oriented [Balci 2000]. Imprecise requirements descriptions must be avoided, as they allow undesired degrees of freedom for modeling. Validation can only be as specific as the documented intended application purpose. Verification techniques can be used to ensure correctness of the model even without a well-defined intended purpose, but also in this case a precise specification of the intended purpose usually helps focusing the verification effort. To assume the availability of a complete and precise specification of the intended purpose in the beginning of the modeling process is an extreme idealization, as usually during model development new system knowledge is gained, which influences the requirements specification retroactively [Davis 1992], but there always should be a current version of the intended purpose specification, which can be used for validating the current version of the Conceptual Model, Formal Model, or Executable Model.

3.1.3 Sufficient System Knowledge

A model can only be validated to the degree of real system knowledge (including data) that is available, or in other words: A validation statement is only as reliable as the knowledge or data it is validated against. In the best case, the real world entities and interactions of the real system and its subsystems are known, and quantitative measures for them exist, which can be measured at the real system as desired. Whenever there is only limited data about the real system, the data only has been or can only be measured at a related system, there are no quantitative measures, some of the dependencies within the real system are not yet explored or modeled, or entities have only abstract counterparts, then meaningful validation of the overall model becomes more difficult. Substantive validation may be constraint to parts of the model, or – in the worst cast – be simply impossible. This prerequisite is the main motivation of section 4.1.

3.1.4 Sufficiently Unbiased V&V Implementers

As motivated by [Arthur and Nance 2000], depending on the degree of subjectivity of the applied V&V techniques and the required reliability of V&V activities, their implementers should be sufficiently un-biased. With respect to software testing, [Myers 1979] states that the search for errors is a "destructive" process, which cannot be efficiently conducted by the one who constructed the item under examination.

- If a member of the developer team conducts V&V, it is assumed that this person is highly biased what most probably leads to wrong (too good) V&V results when applying subjective V&V techniques (see section 4.5). The organizational advantage of *in-team V&V* is obvious – no information or knowledge about the model has to be transferred to a third party and no learning time for understanding the model is required. However, for the purpose of assessment, in-team V&V is only acceptable, if objective V&V techniques with clearly defined quantitative measures are used.
- If there is an in-house quality assurance department with sufficient distance to the project itself it can be assumed that there is no high bias with respect to the M&S project itself, but this is not ensured. Again, the organizational advantage is the opportunity of very close cooperation between the developer team and the *in-house V&V* team, and history of software engineering teaches that this approach leads to higher quality products. But without "hard numbers" from objective V&V techniques, again the reliability of the V&V results for the purpose of accreditation must be doubted.
- It can be expected that an *external* institution is un-biased, but now the communication and information exchange between the developer and the V&V team become more difficult. Intermediate products that are exchanged between these parties must contain all required information. Organizational, psychological, legal, and technical problems must be solved when doing *external V&V*, but only external V&V leads to unbiased or "objective" results when applying subjective V&V techniques.
- If the external institution can demonstrate that it is technically, financially, and managerially independent from the developer [Rosenberg 2001] and does not benefit from either outcome of the V&V process, this external V&V may even be called *independent V&V*.

3.1.5 Minimal Degree of Subjectivity

V&V techniques can be classified ranging from "highly subjective" to "nearly objective". The opinion of a Subject Matter Expert (SME) may be of high value especially for the validation of a model, when the SME compares a mental model of the real system and the expected behavior with the symbolic description of the Conceptual Model and the interpreted, visualized behavior of the Executable Model (see also 4.5.2). However, expertise is hard to judge and often limited to a specific area [Possner 1988], or the SME may be biased for any reason, which results in making (deliberately or not) a wrong statement.

Nearly objective techniques as, e.g., statistical methods for the comparison of model output and system data, or formal methods for constraint checking are based on a mature mathematical background, and are much less dependent on the person applying them to the model. However, the high degree of specialization of these techniques limits their application possibilities and allows only the detection of small ranges of errors. If one prefers to get reproducible, nearly objective results, the use of nearly objective techniques is highly recommended. But for judgment of suitability with respect to the intended purpose, human reasoning is still unavoidable. This implies that for V&V an adequate mix of "highly subjective" and "nearly objective" V&V techniques is required.

3.1.6 Unfeasible Proof of Suitability and Correctness

In practice, there is neither absolute verification nor absolute validation of a whole model. Validation is performed with respect to the real world, which the author assumes is never completely describable in all its complexity. Reality is represented by theoretic assumptions and (mental) models, or by measured data of related systems, real subsystems, or the real system, as discussed in section 4.1. There is no "perfect" or exhaustive knowledge of reality, and thus validation of models and simulation results can only be the failed attempt of systematic falsification.
Verification always requires a precise specification or description of a model, which is transformed into another specification or description following given rules. Through verification one checks, whether the transformation, specification, or description was performed according to the rules, or not. Theoretically it is possible to do so for the complete model, but in practice the model complexity or lack of knowledge about the model prohibit exhaustive verification. For these reasons V&V will not guarantee the correctness and suitability of a model, it remains unproven, whether a residual error or inadequacy remains.

3.2 Existing Related Standards, Guidelines, Methods, and Procedures

In the defense area, for quality assurance of both software, in general, and credibility judgment of models developed for computer-based simulation, already several standards, guidelines, methods, or procedures exist. However, all of them are somehow limited or do not support the achievement of the goals of V&V identified in section 3.1. In the following, a short overview of related work and the basic ideas behind each of the presented approaches is given, followed by a discussion of their potentials, limits, and risks. There is no claim for completeness of the below summary. Only those standards, guidelines, methods, and procedures are listed that directly impacted the approach taken in this thesis.

3.2.1 The Capability Maturity Model for Software

As the creation of software often is an integral part of the development of a computer-based simulation, here a de-facto-standard that is supposed to allow a statement about the expectable quality of a software product is discussed. (However, model development does not equal software development – see also section 2.1.1) The Capability Maturity Model for Software (SW-CMM) was created at the Software Engineering Institute (SEI) at the Carnegie Mellon University in Pittsburgh and was first published in 1990. Here, version 1.1 is analyzed [Paulk et. al. 1993]. The SW-CMM defines five maturity levels based on key attributes that identify an organization at a particular maturity level, and key processes that an organization must have established at a particular maturity level. During an assessment that usually lasts one week, a group of experts determines, whether the requirements defined by each level have been achieved (or not), and assigns according to the results of the assessment an SW-CMM level to the department or organization.

A high level description of the SW-CMM levels as shown in Figure 4 is given below:

- Level 1, “initial”: There are no requirements concerning the software development process. If there is a software development process in the organization, it is only existent on paper, software development is chaotic. No statement about the expectable quality of a software product can me made. If the organization produces high quality software, this is the result of the motivation and qualification of several individuals, but not of quality management.
- Level 2, “repeatable”: Some task areas are clearly defined. An organization on this level has established basic project management. The discipline in applying the process is sufficiently high to repeat successes in similar following projects. Time tables and budget planning are no pure lottery any more.
- Level 3, “defined”: Management truly controls the development of the software and is clearly informed about the advancement of development. There is a software development process that is known throughout the organization and that actually reflects the software development practice in the organization or department. Frequent peer-to-peer reviews are typical in a SW-CMM level 3 organization.
- Level 4, “managed”: An organization at this level has measures at hand to determine both the quality of its software development process and the difference in quality of several project not only qualitatively, but also quantitatively.
- Level 5, “optimizing”: The existing process structure continuously improves itself. Quantitatively measured feedback from the development process allows precise assessment of the achieved degree of improvement. This can be achieved by the continuous application of innovative technology and ideas.

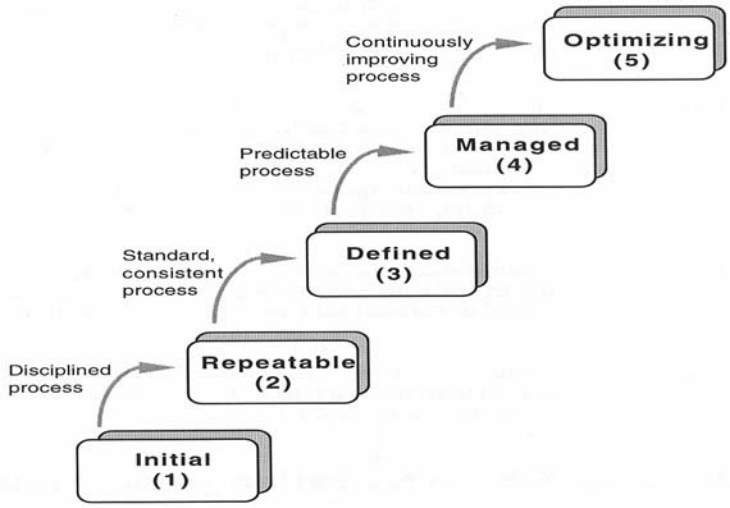


Figure 4: Sketch of the SW-CMM levels [Paulk et. al. 1993]

In the explaining text it is assumed that an enterprise needs approximately a time period of two years for the achievement of the next higher SW-CMM level. Skipping of a level most probably will fail, as the amount and complexity of the processes to be established is assumed not to be manageable.

Transferability to V&V of models and simulation results: The SW-CMM describes a way of introducing commonly as reasonably accepted processes into development, project management, and customer service, and associates the expectable quality of the software product with the maturity of the established processes in these areas. It describes the aims that have to be achieved for each level, but does not rule the way of how to do so. A general increase of software quality can be expected from the use of a de-facto-standard as the SW-CMM.

Each SW-CMM level implicitly contains a statement about the expectable quality of a software product developed by the organization, but does not allow any conclusion for the actually achieved quality of a specific piece of software. If one is going to rely on one unique software product, a general statement about the expectable quality of the software of the organization is not sufficient. However, [Conwell, Enright, and Stutzman 2000] point out the positive impact of well organized software development on the successful execution of V&V of models or simulation results.
The SW-CMM is applicable to software development, which is only one part of model development. It does not require the establishment of a model development process and should be adapted accordingly, if used in the context of M&S V&V.

3.2.2 ISO 9000

The standards or norms family ISO9000 was first published in 1987 by the International Standards Organization, Genf. The most popular norm of this family is the ISO9001, which is a standard for certification of the quality management system of an organization. In the following, the so called "long time revision" ISO9000:2000 as described in [Thaller 2000] is discussed.
Similarly to the capability maturity model for software, the norm focuses on the processes established, but abstains from identifying several levels according to the maturity of the processes. Whether an organization conforms to ISO9001 is decided after the review of the quality management handbook and an audit of two days duration. The requirements documented in the norm are based on the assumption that the quality of a software product heavily depends on the focus on the customer, the management structure of the enterprise, the integration of all involved, the underlying processes, the management system, the continuous improvement, facts based decision making, and a well managed supply chain.
For the purpose of certification, a "quality management handbook" must the filled in. The mandatory contents of the handbook are precisely defined and sketched in the following:

- Quality management system: The norm defines general requirements for the quality management system. Their settlement must be documented according to the documentation requirements.
- Responsibility of management: The responsibility of the management is documented in the handbook. It is set out in writing that management fully supports the introduction of the norm, the organization's work is customer oriented, a well-defined and adequate quality policy will be established, that there is a plan how to do so, that responsibilities and authority are clearly and unambiguously assigned, that the communication between all parties involved works, and that management will control the compliance with the desired quality policy.
- Management of resources: Well-founded statements about the availability of all required resources are found in the handbook. Additionally it is documented that all employees are sufficiently high qualified, the local infrastructure supports product development, and that the working environment is adequate.
- Product realization: The processes of planning the implementation, communication with the customer, design and implementation, sales, production, service, and the control of monitoring devices and measurement tools must also be documented in the handbook.
- Measurement, analysis, and improvement: The organization documents its way of continuous self-control. The methods of quality control and evaluation of both internal processes and internal products, of control of non-compliant products is done, and improvement methods are presented.

Transferability to V&V of models and simulation results: The ISO9000 regulates similar areas as the capability maturity model for software does. The satisfaction of the norm requires structured, well-founded methods and techniques of software engineering and business management, which implicitly leads to exhaustive documentation of all conducted activities. No regulations, which process representation to apply exist, this choice is left to the organization. A general increase in software quality can be expected.
Similar to the SW-CMM as a norm for the certification of organizations, the ISO9000 does not allow a statement about the actual quality of one specific software product. Most of the regulations for software development should be replaced by regulations for the development of computer-based simulations, if this norm was meant to guide M&S V&V.

3.2.3 The V-Model

The "Entwicklungsstandard für IT-Systeme des Bundes – V-Modell" was published first in 1991, and is available today in its version from 1997 (V-Modell97) [Bundesministerium für Verteidigung 1997]. It regulates all activities and products, as well as logical dependencies during maintenance and modification of IT systems, which distinguishes it from ISO9000 or the SW-CMM. In contrast to these, it does not only require the establishing of processes for development, configuration management, quality assurance, and project management, but actually defines precise processes and products in these areas. The development of hardware components is not regulated, but the integration of hardware and software components to an overall IT system is considered. The V-Model complements the above discussed (quasi) standards SW-CMM and ISO9000, as it implements several of the required processes.
The V-Model is split into the four sub-units "project management (PM)", "system development (SE)", "quality assurance (QS)", and "configuration management (KM)". SE is the center with all other sub-units arranged around it. For each sub-unit there is a product flow description, which is synchronized with the product flow descriptions of the other sub-units, and product templates that define the contents of the products clearly and unambiguously. To avoid unnecessary documentation overhead, the mandatory products can be adjusted during SE according to the requirements of the project (tailoring). The sub-units regulate the following:

- In SE a complete software development process with phases, sub-phases, and products is presented, which focuses on system requirements analysis, system design, software/hardware requirements analysis, software high-level design, software low-level design, software implementation, software integration, and transition to use. For each of these there is a phase in the development process assigned to, which includes several sub-phases each, not further discussed here.
- QS describes constructive (preventive) and analytic activities to increase the quality of the software, and regulates the influences of QS on SE. The main activities in the sub-unit QS are the initialization of QS, examination preparation, process examination, product examination, and QS reporting.
- The sub-unit KM regulates all activities for managing the different versions of the product in a structured form. The four distinguished phases are KM planning, product and configuration management, modification management, and additional KM services.
- The sub-unit PM regulates all organizational aspects. Explicitly considered are the project initialization, project acquisition, internal management, detailed planning, cost-benefit-analysis, realization decision, risk management, project control, reporting, teaching, resources management, sub-contract management, employees briefing, and project completion.

Transferability to V&V of models and simulation results: The V-Model contains a precisely defined SE process, which can serve as basis for integration of V&V activities after addition of M&S specific elements. Exhaustive quality assurance (QA) activities are scheduled, which may be implemented using V&V techniques.
The main focus of the V-Model is on the sub-unit SE and software development, as the phases SE3 through SE7 describe requirements analysis, design, and implementation of software. Questions characteristic for model development and the design and implementation of the highly specialized simulation software are only considered marginally.

3.2.4 NASA Likelihood of Failure Rating

The NASA developed a categorization scheme for the likelihood of failures in a software project, to provide an objective basis for the decision, whether independent V&V (IV&V) is required, or not [Rosenberg 2001]. Indicators for the likelihood of failure are so called *factors*, as there are:

- the complexity of the software team (including team size and team member backgrounds),
- the degree of support by sub-contractors,
- the complexity of the organization, including geographical distribution and communication infrastructure,
- the time pressure,
- the SW-CMM level of the organization,
- the degree of innovativity of the software,
- the complexity of integration, especially for distributed software,
- the degree of maturity of the requirements specification, and
- the number of lines of code.

Depending on weighted factors that influence the development of the model, one estimates a "likelihood of failure" rating. Each of the above influence factors is quantified in five levels 1 – 5 with an unweighted *probability of failure score* of the value 2^{level} assigned to it. These failure scores of each factor (bullet list above) are weighted single or double, according to their assumed significance. (Weighting factors are fix.) The sum of all weighted failure scores yields the *likelihood of failure rating*.
In a second step, the necessity for IV&V is determined using another categorization scheme where consequences of software errors are classified as "insignificant", "marginal", "substantial", and "grave". Depending on the consequence class and the calculated likelihood of failure rating, the table states, whether IV&V is required, or not.
Transferability to V&V of models and simulation results: This approach seems to be suitable for the determination of the required rigor and intensity of V&V. However, although the calculation of the likelihood of failure rating creates the impression of an objective estimation, the initial scores for each factor level and their weights make it highly subjective. No study results are known that relate the actual numbers of errors found after rigorous V&V (which indicate the "likelihood of failure") with the calculated likelihood of failure rating on an empirical foundation. In this case this is not really relevant, as the rating only supports a binary decision (IV&V or no IV&V), but one must be careful, if this approach shall be transferred to support more subtle decisions.

3.2.5 The DMSO Recommended Practices Guide on VV&A

The Recommended Practices Guide (RPG) of the US Defense M&S Office [Defense Modeling and Simulation Office 1996 and 2000] most probably is the most extensive document in the research area of M&S VV&A currently available. It has been published first in 1996, but several of its numerous authors (Glasgow, Balci, Muessing, Youngblood, Solick) published

their contributions to the US Defense Modeling and Simulation Office in only slightly modified form previously on scientific conferences. The current RPG2000 is a completely modified version, that treats numerous more or less relevant aspects of VV&A. An excellent introduction to the background and necessity of VV&A is provided, illustrated by examples. This includes definition and explanation of the terminology, process models, identification of people or "roles" involved, guidelines for the approximation of cost and benefits, and more. Most of the contents of the RPG2000 are described in a process-oriented way, are easily comprehensible and provide an excellent overview of all organizational problems. Some guidance is provided even down to the selection of V&V activities.
Without any doubt, the RPG2000 describes the current state of the art of VV&A methodology and satisfies its title. Nevertheless, despite (or because) of the extend of this work, there are inconsistencies, and some of its limitations become obvious after closer examination:

- Within the core documents, the usefulness of the differentiation according to the roles within the problem solving process must be doubted. The separation between the guidelines given for each role is imprecise, which leads to frequent repetitions of text pieces over the whole document. One the other hand, information that definitely is of interest for a specific role is hidden in another section of the core document associated to another role. If one is seriously interested in VV&A, one has to read all core documents anyway, else most probably information that is relevant for the own role is missed.
- The RPG2000 is both a technical and a political document. This seems to be the reason, why products, processes, and guidelines are not precisely defined and leave enormous freedom for interpretation. Those who actually are involved in ensuring that a model or simulation results are correct and suitable, find beside a list of uncritically assembled V&V techniques only little practically applicable guidance. Detailed regulations, which working steps should be executed for the V&V of a model or simulation results, and detailed product specifications as given in the V-Model are not contained in the RPG2000. It does not provide guidance through model development that allows the explicit association of QA activities with V&V techniques.

3.2.6 Confidence Levels in Model Behavior

The CLIMB process (Confidence Levels in Model Behavior, [Flight Mechanics Panel Working Group WG-12 on Validation of Missile Simulation 1985]) categorizes comparison data in five levels, according to the source they origin from. This significantly distinguishes CLIMB from other approaches, where accreditation is a binary decision without additional information concerning the underlying V&V techniques, including their rigor and intensity. The approach has been introduced to increase the credibility of simulations used in the context of rocket development for system improvement and capability demonstration.
The requirements for reaching a desired confidence level concern

- the documentation and origin of the comparison data;
- the documentation of the model, runtime data, and model use;
- the necessity to satisfy all requirements of the next lower CLIMB level.

The origin or source of the comparison data are the main characteristics of the desired level. This is sketched in Figure 5.
To reach CLIMB 1, model output must be compared with the expected output of the model based on "educated guesses" (expert opinion). The Conceptual Model is evaluated by an SME, too. The model documentation is expected to include

- a description of the examination aim (intended purpose of model use),
- a rough description of the Conceptual Model,
- a functional model,

- a description of the applicability of the model,
- a comment on its implementation,
- the model history, and
- comments on the behavior of the model.

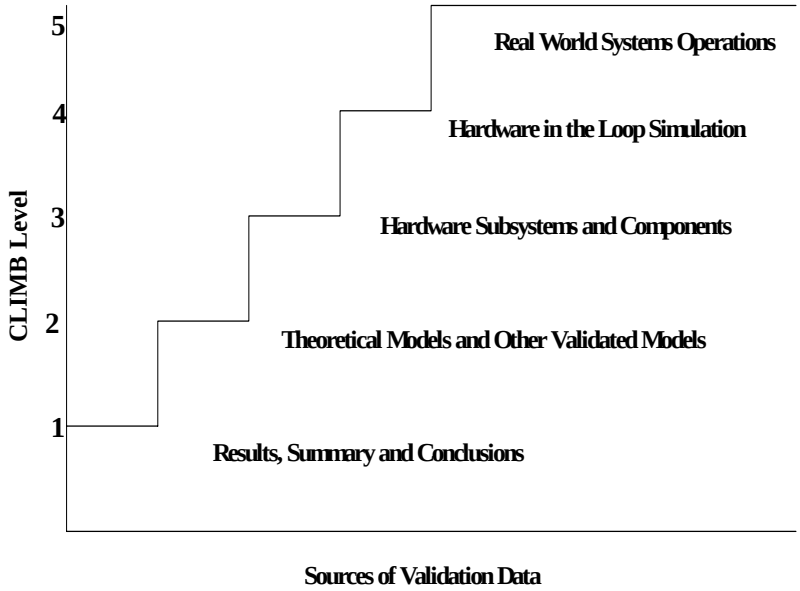


Figure 5: The CLIMB and their comparison data

For CLIMB 2 already a comparison of behavior between the model and another documented (and accepted) theoretical model or another accredited Executable Model is required. In addition it is examined, whether the program specification (design) was met. Additional documentation requirements include

- block diagrams of the Conceptual Model and all of its submodels,
- a basic description of the implementation, and
- a benchmark test run.

In addition, all requirements for CLIMB 1 must be fulfilled.

For CLIMB 3 the behavior of the submodels of the model must be compared to the behavior of their corresponding real subsystems. The comparison data must be measured at the existing real subsystems in the lab. Additional documentation requirements include

- the description of the lab test setup,
- the description of the data measurement method, and
- instructions for the creation of simulation runs.

In addition, all requirements for CLIMB 2 must be fulfilled.

To achieve CLIMB 4, comparison data is required, which was measured at a hardware-in-the-loop simulation that interconnects all already existing real subsystems. Then the results of the "software only" model are compared to the results of the HIL simulation. Additional documentation requirements include

- the description of the HIL experiment setup, and
- the modifications of the model.

In addition, all requirements for CLIMB 3 must be fulfilled.

The highest CLIMB 5 requires the comparison of simulation results with the actually observable behavior of the real system. Additional documentation requirements include

- the description of the real experiment setup and
- the description of the simulation setup.

In addition, all requirements for CLIMB 4 must be fulfilled.

The CLIMB is an approach to define minimum requirements for V&V activities and documentation for a desired level of model credibility. With the requirements for the distribution of comparison data (breadth) and their origin (depth), a (highly specialized) measure for V&V rigor and intensity has been introduced. Those who are familiar with the CLIMB concept know from the achieved level, under which constraints the model was validated. Under the assumption that the V&V activities required for each level were implemented carefully and correctly by qualified personnel, a conclusion about the expected reliability of the model is possible.

CLIMB is a highly specialized approach to increase the confidence in a special kind of M&S. It is most likely to be successfully applied to models of real systems that are under construction. With advance in development, more about the real system is learned, and the (adjusted) model becomes more likely to simulate the (expectable) behavior of the final real system. In the beginning of the development process of the real system, there is only the specification of its desired properties, and the Executable Model may serve as a "dynamic specification" of the real system. Whether the real system that will be developed to meet these requirements will behave as shown by the model is not clear at all. With the completion of the detailed design of the real system the correctness and suitability of the reflection of the structural dependencies within the model can be evaluated, which leads to an increase of confidence (if true). As soon as real data can be measured from the (prototypically) implemented components of the real system, the behavior of their representatives in the model (corresponding submodels) can be validated. Continuing this approach consequently, the real components are interconnected in a virtual system (hardware-in-the-loop simulation), which yields comparison data for the overall model. As the HIL simulation contains many real components, which will be used in the real system it is assumed that its output data is already very close to the data that will be measured after the real system's completion. Finally the completion of the real system allows the direct comparison between model output and real system operation data.

The strength of this approach lies in the stepwise integration of V&V results of the components, which allows to gain confidence in the validity of the structure from pure behavior validation (bottom up black-box and integration testing). With the validation of each integration step of the validated components, indirectly the structure description is also validated. After extensive testing of the components and validation of the corresponding submodels, only a few tests with the completed real system should be required to confirm the correctness and suitability of the real model.

CLIMB is mainly validation-oriented and does not cover the whole breadth and depth of the achievable V&V intensity. The selection of the focus is not amazing at all, as in the context of CLIMB exhaustive tests of the components of the real system and the real system itself, measurements of physically and quantitatively defined measurands are possible. However, the less opportunity for the comparison between simulation results and real data, the more important becomes verification that ensures that the symbolic description of the structure and behavior of the model most likely results in the desired, not directly comparable behavior.

3.2.7 RAND VV&A Taxonomy

A pragmatic approach to the VV&A of models for combat and battle field simulation was presented by [Davis 1992] as research results for the U.S. Under Secretary of Defense for Acquisition. He neither deals with a V&V process, nor with the definition of V&V levels, but concentrates on a *taxonomy for VV&A* on the one hand, and *incremental VV&A* and re-use of

previously achieved V&V results on the other. The following issues are addressed within the taxonomy:

- Verification is split into *verification of logics and mathematics* and *verification of the program code*.
- The attempt to demonstrate validity is subdivided into *empirical evaluation*, *theoretical evaluation*, and *evaluation by other comparisons*. (Empirical evaluation mainly includes comparison against data from history, field-test, laboratory, maneuvers, and other exercises; theoretical evaluation addresses analytic rigor, verisimilitude, clarity and economy, and other validated models and scientific theories; and evaluation by other comparisons uses expert opinion, doctrine, models of uncertain validity, and other information sources.) In addition, in analogy to [Zeigler, Praehofer, and Kim 2000] it is distinguished between predictive, descriptive (replicative), and structural validity, although the definition of structural validity deviates from Zeigler's.
- Model accreditation is divided into provisional accreditation for an application class and accreditation in the context of a particular analysis plan.

Severity Category		CATASTROPHIC	CRITICAL	MARGINAL	NEGLIGIBLE
Ref	**Impact Domain**				
1	**Personal Safety**	Death	Severe injury	Minor injury	Less than minor injury
2	**Occupational Illness**	Severe and broad scale	Severe or broad scale	Minor and small scale	Minor or small scale
3	**System Damage**	Loss of system	Major system damage	Minor system damage	Less than minor system damage
4	**Environmental Impact**	Severe environmental damage (eg. Chernobyl)	Major environmental damage (eg. Most land blight)	Minor environmental damage (eg. pollution of a stream)	Trivial environmental damage (eg. minor spillage with no long term effects)
5	**Operator Workload**	Operator cannot continue to operate system	Severe reduction in the ability of operator to operate system	Major reduction in the ability of operator to operate system	Minor reduction in the ability of operator to operate system
6	**Financial Loss**	Above £1m	£250k to £1m	£10k to £250k	Less than £10k
7	**Security Breach**	Top Secret	Secret	Confidential	Restricted
8	**Reliability**	Total loss of functional capability	Severe reduction in functional capability	Significant reduction in functional capability	Slight reduction in functional capability
9	**Project Schedule**	Slip impacts on overall defence capability	Slip impacts on other projects (eg. life extension of existing system)	Slip results in major internal schedule reorganisation	Schedules republished
10	**Mission Impact**	Mission loss (operational)	Severe mission degradation (operational)	Slight mission degradation (operational) Mission loss (training)	Mission delayed (operational) Mission degraded (training)
11	**Criminal Liability**	Custodial sentence imposed	Large fine imposed (£5k plus)	Small fine imposed (up to £5k)	Conditional discharge etc.
12	**Civil Liability**	Multiple, large civil suits (£10k plus)	Single, large civil suit (£10k plus)	Multiple, small civil suits (up to £10k)	Single, small civil suit (up to £10k)
13	**Maintenance Burden**	Projected servicing schedules severely adversely affected	Unscheduled maintenance predictions severely adversely affected	Projected servicing schedules slightly adversely affected	Unscheduled maintenance predictions slightly adversely affected
14	**Political Impact**	Government falls	Minister resigns	Commons debate/National Press aware	Parliamentary Question/Local Press aware
15	**Delivered System Performance**	Design does not meet requirement in critical areas - leading to a failure to accept system	Design does not meet requirement in non-critical areas - leading to major modification programme	Impact on operating procedures	Some trivial deficiencies

Table 1: Impact domains and severity categories from [Mugridge 1997]

Under consideration of this taxonomy, activities or techniques are identified, how to ensure or assess each type of correctness or validity. It should be recognized that the process of accreditation is handled with great care, allowing only the accreditation of models in the narrow con-

text of an analysis study, or the provisional accreditation for a wider application field (class accreditation). This expresses the (justified) concern that otherwise a model once officially "accredited" will be (ab-) used for applications it is not intended for, with the "accreditation stamp" as legitimation.
When re-using a model in a study, information on previous V&V influences the V&V of the revised model, which is also explicitly stressed, but not developed in more detail. The clear differentiation between empirical, theoretical, and "other" evaluation helps to concentrate the validation effort on the aspects of the real world with sufficient information available. However, dependencies between the identified elements of the taxonomy are not pointed out in detail.

3.2.8 DERA Risk Classification

[Muessing, Laack, and Wrobleski 1997] published a matrix for the classification of severity of the impact of a decision, which has been extended and refined by [Mugridge 1999]. The upper border of the severity of the application of a model or simulation can be estimated giving consideration to the possible worst case impact of wrong simulation results. To unify this estimation, 15 impact domains are given, each with characteristics for the classification into one of the four severity categories. This allows to create a "severity profile", or to determine a severity level. Lacking clearly defined quantitative metrics or empirical data, tables are used for the quantification [Muessing, Laack, and Wrobleski 1997; Mugridge 1999]. Table 1 shows a summary of worst case consequences of a wrong model or simulation results on the real world. This includes a classification of these consequences according to their severity.
The impact domains are weighted, which controls their impact on the *required level of credibility*. The required level of credibility then drives the selection of V&V activities taken from another table given in the technical report.

3.2.9 Balci's Evaluation Environment

To support credibility assessment, Balci developed an evaluation environment for models and simulation results. Based on the postulation that modeling is an art and credibility assessment is situation dependent, [Balci 1990] justifies peer assessment as a quite effective method for evaluating the acceptability of simulation results. The peer panel, created from experts knowing the system under study, expert modelers, expert simulation analysts, and simulation project experts, evaluate all available information about the model and the simulation results, based on indicators at the leaves of a predefined indicators hierarchy. To each indicator a score out of 100 is assigned, expressing the reviewer's confidence that the requirements identified by the indicator are met. Each indicator is weighted according to its importance. If required, paths can also be weighted to reflect their meaning in a particular simulation study. Figure 6 depicts a sample indicators hierarchy.
Both advantages and disadvantages of this approach seem quite obvious. The predefined indicators hierarchy provides excellent guidance for planning and execution of V&V activities, and structures V&V results. However, as [Sargent 1999] clearly states, it is hard to map credibility statements for each indicator to scores of 0 – 100, to justify the chosen weights for the propagation of the indicator scores, and to draw any conclusion for the overall acceptability of simulation results from the propagated score. A high degree of subjectivity is hidden in an objective looking approach. However, the approach was continued and the Evaluation Environment stepwise improved. Its current state is documented in [Balci 2001].

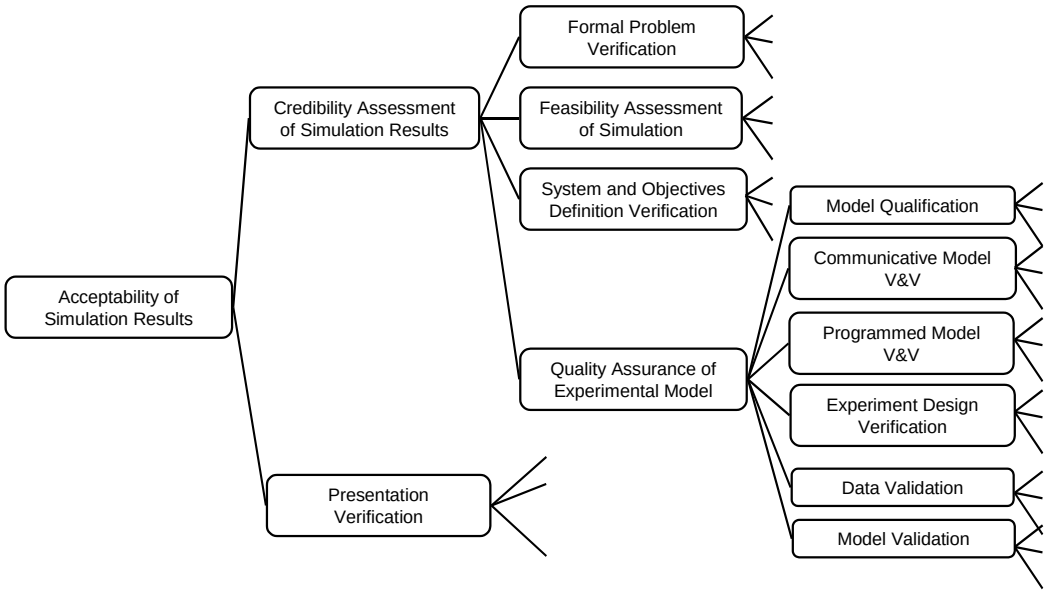


Figure 6: Indicators hierarchy from [Balci 1990]

3.2.10 The International Test Operations Procedure on V&V

The V&V International Test Operations Procedure (ITOP) currently is under construction. Its first release to a selected reviewers community is expected for autumn 2003 [International Test Operations Procedure on V&V 2003]. Numerous issues addressed in this thesis influenced the development of the ITOP on V&V. It is a high-level document intended to provide a standard to support the exchange of V&V information among the ratifying nations France, Germany, the United Kingdom, and the United States of America. It comprises a procedure and guidance for planning, implementing, and documenting V&V efforts of models and simulations. The intention is to make M&S results acceptable, usable, and re-usable to the intended nations. For this purpose it provides

1. modular cases for separate V&V of the model, the data, and model use, to support the clear separation of the model itself, embedded and runtime data, and the simulation scenarios;
2. a structured approach for organizing claims, arguments and evidence as a basis for an informed accreditation decision, which is based on a fixed indicators hierarchy flexibly extendable for each individual V&V effort [Pygott and Wilson 1996];
3. a framework of levels addressing the level of impact, level of V&V, and level of quality;
4. a workbook template for the uniform documentation of the modular cases, which contain the claims, arguments, and evidence.

Claim-Argument-Evidence Structures (CAE structures) support the structured presentation of a chain of arguments that motivate the claim for the correctness and suitability of a model or simulation results. The overall claim is hierarchically and flexibly decomposed into subclaims, with an argument justifying the decomposition. It is not intended to define propagation rules through the indicators hierarchy. The satisfaction of the lower claims will, in turn, lead to the satisfaction of higher claims until the highest claim of M&S credibility is reached. A discussion about the potential and limits of the ITOP on V&V is found in [Lehmann et. al. 2002]

3.3 Summary of Basic Concepts

The analysis of the related work summarized in section 3.2 resulted in the identification of several basic concepts of V&V. Some of them differ significantly, others are very similar. The identified basic concepts, which need to be considered for the specification of a V&V framework following in chapter 4 are discussed in the following.

3.3.1 Process Approach

As shown in section 3.2.1, 3.2.2, and 3.2.3, and according to principle 3.1.1, numerous published approaches to software quality assurance (QA) or V&V of models and simulation results require or are based on a process model, where phases of the model development process are overlaid by V&V phases, e.g. [Defense Modeling and Simulation Office 1996 and 2000; Graffagnini, Youngblood, and Lewis 1999]. Another commonly accepted approach are feedback loops within the development process with V&V activities linked to them [Sargent 1999]. In general, these approaches can be distinguished by the selection of the type of the model development process and the assignment of V&V activities to the corresponding phases or feedback loops, as discussed in section 1.4.
The advantage of the process approach is that there is explicit guidance how to proceed for model development and V&V. However, this advantage may become a disadvantage, if V&V activities have to be harmonized and synchronized with a model development process already established in an organization.

3.3.2 Question Catalogues and Fixed Indicator Hierarchies

The approaches presented in section 3.2.9 and 3.2.10 feature question catalogues and fixed indicators hierarchies. Checklists in general aim on excluding expected mistakes by explicitly checking, whether a specific property has been satisfied (or not), which showed to perform well in the area of software engineering (e.g., Fagan's Inspections, [Office of Safety and Mission Assurance 1993]). These questions can be pure lists of properties to check within or without a recognizable order. Alternatively, the overall question of correctness and suitability of the model may be hierarchically decomposed, yielding a fixed or static indicators hierarchy. Each indicator is decomposed into a set of several more precise indicators. During V&V the indicators at the leaves of this hierarchy tree are examined, which allows to draw conclusions for the indicators on the upper level. Examples of these approaches are given in section 3.2.9 and 3.2.10, additional reference include the Cost Estimation Tool of Lewis [Kilikauskas et. al. 2002], TECOM Pamphlet 73-4 [Department of the Army 1999a], or the Army Pamphlet 5-11 [Department of the Army 1999b].
Several advantages come along with this approach:

- The choice of questions defines the width of examination (see also section 4.1.2);
- Those responsible for V&V can choose, whether to approach the problem bottom-up starting with the leaves of the hierarchy, or top-down starting with the root;
- The examination applied to different models or simulation results is directly comparable, when the same checklist or indicators hierarchy was used.

However, the following limitations can be identified:

- The width of V&V is anticipated by the (inflexible) selection of the indicators. It is unlikely that any given questions catalogue below a certain degree of abstraction is complete for any given model or simulation results. Therefore, the existence of loopholes must be assumed.
- The indicators hierarchies suggests the assignment of credibility values and their propagation through the hierarchy. There are serious limitations to this approach, as discussed in [Sargent 1999].

3.3.3 Flexible Indicator Hierarchies

In the approach presented in section 3.2.10 also an indicators hierarchy is created, but in contrast to 3.3.2 its structure is not predefined but adapted for each V&V effort individually. This implies that in addition to the indicators hierarchy itself the rationale for its creation must be documented, too.

Several advantages results from flexible indicators hierarchies: The precision of the questions can be increased or decreased by individual, aim-oriented hierarchical decomposition or abstraction, i.e., it supports both a flexible bottom-up and a top-down approach. Starting bottom-up, for all available V&V evidence (products generated as outputs of V&V activities) claims are created, which they support, and these claims are *flexibly* connected by an argument to a claim at the next higher level in the hierarchy. Alternatively, starting top-down, the overall claim for correctness and suitability is decomposed *flexibly* into sub-claims, which are again decomposed into sub-claims, until sub-claims are identified for, which V&V techniques and sufficient information are available. Then the V&V results are created, which serve as evidence. In practice and especially for model re-use and evidence leveraging, a meet-in-the-middle approach seems to be most appropriate.

However, the case dependent selection of arguments can also become a drawback. The identification of indicators and the reasoning of the argumentation must be reviewed with each new V&V project – in contrast to the fixed indicators hierarchy discussed in section 3.3.2. A combination of these two approaches seems to be reasonable, with a fixed hierarchy for the more abstract indicators, and a flexible, case sensitive claim-argument structure when it comes down to the features of the model.

3.3.4 Risk-Orientation

Directly or indirectly addressed in the approaches taken in the sections 3.2.4, 3.2.6, 3.2.8, and 3.2.10, this concept deals with the determination of the required V&V effort for a specific intended use of the model, depending on the worst case impact of a wrong simulation-based decision and the probability of a wrong model or invalid simulation results. Usually in risk-oriented approaches to V&V, it is distinguished between impact of the decision, influence of the model or simulation results on the decision, and the probability of an unsuitable or incorrect model or simulation results. State of the art is the use of tables to come up with a conclusion about the required rigor and intensity of V&V. Theoretically, these factors can be arithmetically connected as discussed later in section 4.1, or in [Harmon, Gross, and Youngblood 1999]. However, as long as there are measures or metrics for neither impact nor influence, these considerations are purely academic.

3.3.5 Process Assessment vs. Product Assessment

The processes presented in section 3.2.1 and 3.2.2 concentrate on the assessment of the development process followed (“process-oriented”), while those in sections 3.2.6 and 3.2.10 concentrate on the (intermediate) products generated during development (“product-oriented”). The question, whether V&V shall be more process-oriented or more product-oriented remains unresolved in the V&V community; to motivate the selected orientation for this thesis, both tendencies are explained and their advantages and disadvantages briefly discussed:

- *Process-oriented V&V* focuses on the evaluation of the activities conducted during model development and V&V. It is the aim to ensure that the model has been developed in a reasonable way in accordance with a well-defined model development process, using sound modeling and Software Engineering techniques. This approach (which is also the idea of ISO9000 [Thaller 2000] or the SW-CMM [Paulk et. al. 1993]) is based on the assumption that a careful and structured approach leads to the creation of a high-quality product. It is not necessary for the model developer to produce additional information that supports the evaluation of the correctness and suitability of the model in addition to the products re-

quired within the certified process. It is sufficient to demonstrate that model development was conducted according to the process. This minimizes the additional effort for the model developer and leads to products of higher quality in general, but does not allow to judge the correctness and suitability of one particular model or a particular set of simulation results. As models are no mass product and often each individual model is of great importance, for the V&V of a specific M&S product pure process-oriented V&V seems to be hardly acceptable.

- *Product-oriented V&V* focuses on the assessment of the correctness and suitability of a specific product. The product itself is subjected to examination, independently from the way it was created. Pure product oriented V&V allows assessment of the product quality without constraining or limiting the developer with mandatory development guidelines. However, strict guidelines for the documentation of all intermediate products gained during model development are required, as completeness and understandability of the model documentation are essential for this approach. This leads to a very high effort for product oriented V&V, which on the other hand also enhances reuse of the model.

The main focus of scientific literature lies on the evaluation of developmental products (product-oriented V&V) [Balci 1998b; Sargent 1999; Pace 1999b]. Due to pragmatic reasons, some authors now tend towards process-oriented V&V [Balci 2002]. A combination of both approaches promises to be sufficiently precise and manageable, if it focuses on the product for all critical aspects.

3.3.6 V&V Degrees or V&V Levels

In the approaches presented in sections 3.2.1, 3.2.4, 3.2.6, 3.2.8, 3.2.9, and 3.2.10, levels have been introduced to either express intensity or rigor of the conducted required V&V activities. Under the assumption that the proof of correctness not always, and the proof of suitability only rarely can be given, the V&V level or degree contains a statement about the expectable residual error or the allowed confidence in the model or simulation results.
The advantage of using well-defined levels or degrees is clear; the binary statement about correctness and suitability is replaced by a refined statement, which expresses the degree of perceived correctness and suitability. Ideally, this removes the implied unreliability of each V&V statement. However, as there are no commonly accepted metrics for measuring the reliability of V&V techniques at the current state of the art, the (subjective) weighting of V&V results returns the uncertainty to the statement, and their significance must be doubted.

4 THE GENERALIZED V&V-PROCESS

In the previous chapters the need for a V&V process for models and simulation results that provides detailed and consistent guidance for effective and efficient planning, implementation, and documentation of V&V objectives, activities, and results was identified. In this context, V&V shall be a comprehensible, risk-oriented, and manageable quality assurance measure that is applied throughout the whole model development process. Its results shall be documented in a form that supports assessment of both the rigor and intensity of the conducted V&V. The new V&V process, which is developed in this thesis shall be distinguishable from existing guidelines by

- identification and motivation of **clearly defined objectives** that shall be achieved during V&V in parallel to model development;
- precise specification of all required **products (inputs)** from model development and from other sources of external information, to achieve these objectives;
- detailed guidance on **how to choose and use V&V techniques** to achieve the V&V objectives,
- a clear description of the **expectable V&V results (evidence)**,
- the clarification of the **dependencies** between the V&V objectives, and
- detailed guidance on how to **reuse previously created V&V results** in subsequent stages of V&V.

This section discusses the external influences, drivers, and boundaries of V&V, and derives the design for a generalized V&V process. As the most important factors, which influence the planning, implementation, and documentation of M&S V&V, the following are identified:

1. **Required rigor and intensity of the V&V results**, which depends on the risk associated with the use of the model or simulation results, which impacts the likelihood of residual errors,
2. **Real system knowledge**, i.e., availability of information (theoretical and practical knowledge and "hard" measured data) about the real system, which limits the maximum possible intensity of V&V, and
3. **Model knowledge**, i.e., availability of information about the model or simulation results themselves, which also limits the maximum possible intensity of V&V.

To control these influences, the goals, constraints, and requirements discussed in previous chapters suggest that the M&S V&V process is embedded into a set of several processes:

1. A **risk classification process** that helps to appreciate the minimum required V&V effort and focuses the identification of V&V objectives.
2. A **model development process** with well-defined intermediate products that structures the development of the model and guaranties the existence of the minimum required information about the model.

3. A **credibility building process** during which the insight that was gained and documented during the V&V process is evaluated.

This chapter first concentrates on the factors that influence the planning, implementation, and documentation of M&S V&V. Then the credibility building process is discussed, and the V&V process outline is sketched. Finally the developed V&V process, which is specified in detail in Appendix A, is introduced.

4.1 Application Risk and Required V&V Effort

The main driver for the V&V of models or simulation results is the risk incident to their application [Kilikauskas et. al. 2002; Harmon, Gross, and Youngblood 1999; Muessing, Laack, and Wrobleski 1997; Mugridge 1999]. Simulation results must only be used, if they are sufficiently credible with respect to the impact of their use, and the influence of the simulation results in comparison to other non-M&S influences ("conventional" information). If the influence of the model, simulation results, or observed model behavior is high, wrong or unsuitable simulation results are not compensated by conventional information, and most probably lead to wrong decisions with undesired consequences. For example, wrong behavior learned in a training simulator that is not compensated by other training methods may lead to severe damage or loss of the real system during its operation (including human death). Although the following discussion results in the statement that it is currently not possible to compute a significant quantitative value for the risk associated with a simulation study, it helps to clarify the dependencies between the distinct influences on risk.

4.1.1 General Approach to Risk Assessment

Several parameters of the risk associated with a simulation-based decision are summarized below:

- Models or simulation results themselves are harmless, as long as they do not initiate any action that impacts the real world. They only become counterproductive or even dangerous, if they suggest a *wrong or erroneous decision* for the real world with undesired counterproductive consequences. Let *E* be the (undesired) event that a wrong decision is made ("erroneous decision").
- The *worst case impact I(E)* of the wrong decision may be directly derived from the area the decision affects, and is completely independent from the nature or origin of the information base used for decision making. If the decision may, for example, result in human death or massive financial loss, the impact of the decision is considered to be high. Otherwise, if the injury or financial loss is negligible, the impact is low [Muessing, Laack, and Wrobleski 1997; Mugridge 1999]. As an externally provided constraint, *I(E)* is considered to be constant with respect to V&V, as the quality of the simulation results does not influence the worst case impact at all. (The determination of the severeness of the worst case impact *I(E)* of a wrong decision is not up to those involved in developing, verifying and validating the model or simulation results, but to the user of the model or simulation results.)
- The probability of making a wrong simulation-based decision $P_{Sim}(E)$ depends on the correctness and suitability of the information base on which the decision is made.

Following a typical interpretation of risk, the risk R_{Sim} incident to a simulation-based decision is defined as the product of the probability $P_{Sim}(E)$ of making the wrong simulation-based decision *E* and the worst case impact *I(E)* of the wrong decision. (As unit for *I(E)* and R_{Sim} a currency may be used.) This can be expressed as

$$R_{Sim} = P_{Sim}(E) \cdot I(E) \tag{1}$$

The information base used for decision making is completely or partially gained by the use of models or simulation results. Let O_E be the event that the *simulation results* on which the decision is based *are erroneous*. From the perspective of those involved in M&S, the probability $P_{Sim}(E)$ of an wrong simulation-based decision can be decomposed in

(1) the probability $P(O_E)$ that the simulation results actually are erroneous, and
(2) the probability $P(E|O_E)$ that wrong simulation results O_E lead to a wrong decision E. ($P(E|O_E)$ expresses the influence of the simulation results.)

This can be formulated as

$$P_{Sim}(E) = P(E \mid O_E) \cdot P(O_E) \tag{2}$$

Depending on the share of simulation results on the whole decision base, the correctness and suitability of the simulation results influence $P_{Sim}(E)$ more or less. I.e., if the simulation results are only marginally considered during the decision making process, their influence is respectively low, which can be expressed by a low value of $P(E|O_E)$, close to zero. If the decision is nearly completely simulation-based, the influence of the simulation results is high, resulting in a high value of $P(E|O_E)$, close to one. If other sources of knowledge are consulted besides simulation, the influence of the simulation results depends on their weight during the decision making process.

Treating the share of simulation-based knowledge of the decision base as an external requirement, $P(E|O_E)$ is not influenced by V&V and therefore constant. This clarifies that V&V excusively reduce the probability of negative influences of a model or simulation results on the overall decision making process by reducing $P(O_E)$.

Finally, the decision maker has to accept a residual risk R_{Max} when making a decision. Again, the acceptable residual risk R_{Max} is treated as a constant with respect to V&V, as V&V does not modify it. If $R_{Max} \geq R_{Sim}$, i.e., the risk R_{Sim} of making the decision on the available information base is less than or equal to the maximum acceptable residual risk R_{Max}, the model or simulation results are acceptable for use. Using equations (1) and (2), this can be formulated as

$$R_{Sim} = P(E \mid O_E) \cdot P(O_E) \cdot I(E) \leq R_{Max} \tag{3}$$

Unfortunately, due to the lack of clearly identified and commonly accepted measures of model quality, today it is not reasonably possible to estimate or even compute $P(O_E)$. [Rosenberg 2001] gives a first rough guess of $P(O_E)$, but without any empirical foundation (see also section 3.2.4). $P(E|O_E)$ most probably also depends on the background of the decision makers, and thereby is highly subjective. Nevertheless, the following qualitative conclusions can be drawn from or confirmed by the above discussion:

1. The lower the acceptable probability of erroneous simulation output $P(O_E)$, the lower must be the maximum acceptable residual risk R_{Max}.
2. The maximum acceptable probability of erroneous simulation output $P(O_E)$ decreases with increasing influence of simulation results on the decision $P(E|O_E)$.
3. If the probability of erroneous simulation output $P(O_E)$ cannot be reduced below a given threshold, the decision makor (often also the sponsor) needs to deliberately reduce the influence of simulation results on the decision, or change the intended use of the model to decrease the impact of an erroneous decision $I(E)$. Otherwise a higher maximum residual risk R_{Max} must be accepted.

4.1.2 Required V&V Rigor and Intensity

The minimum required *rigor* and *intensity of V&V* (see section 1.5.4) is implied by the desired credibility, which directly depends on the risk R_{Max} acceptable for the intended model use. Without being able to give a reasonable estimate of $P(O_E)$, it is an even more desirable goal to reduce the probability of erroneous simulation output $P(O_E)$ by increasing the intensity of V&V. The desire to give an at least rough estimate of V&V intensity leads to the differentiation between the *depth* and the *breadth* of V&V. The V&V intensity depends on "what was checked" (breadth) and "how intensively it was checked" (depth). Both achievable depth and achievable breadth of V&V are limited by the available system and model knowledge.
Breadth is associated with rigor and describes the variety of objectives achieved during V&V concerning the previously introduced intermediate products. Indicators for the breadth of the conducted V&V are:

- the variety of types of model information (intermediate products) evaluated, i.e., aspects of the model, which were examined, and their associated documentation;
- the variety of information about the real system evaluated, including comparison data and comparison data sources (from both models and reality), and theoretical and practical knowledge (including the number of SME, and the number of domains where the SME came from);
- the variety of V&V techniques applied and documented.

For example, to increase rigor, after conducting V&V of the Structured Problem Description and the Conceptual Model, one may instruct an expert to verify the Formal Model for syntactical and semantic correctness (see section 5.1), and to use statistical techniques to compare model results to test data from the real system (see section 6.5.2).
The *depth* describes the intensity of examining the different model contents. Indicators for the depth of the conducted V&V are:

- the concentration of analysis and testing on one particular intermediate product, and the level of detail of the documentation evaluated;
- the density of comparison data (from both model and reality) and special knowledge about the real system (e.g., reputation of the SME consulted on a specific topic);
- the maturity and objectivity of V&V techniques applied and documented.

For example, to increase intensity, after behavior visualization of the Executable Model and its evaluation by an SME (face validation, see section 5.4), its syntactical and semantic correctness can be shown by a syntax and semantic checker (see section 5.1), and model behavior is evaluated by using an additional mature V&V technique like model checking (see section 5.3.7).

4.1.3 Risk and Credibility Classification

As a conclusion from the above discussion, it can be summarized that a scalar measure for risk as provided by the risk classification process of [Mugridge 1999] is helpful as an overall risk statement, but does not support the focused identification of V&V objectives. This requires at least a statement of *prioritized worst case impacts* to allow the identification of their potential causes within the model or simulation results and the concentration of V&V activities on them. Another discussion on V&V metrics can be found in [Pace 2003].
As it is not the intention of this thesis to offer a solution for the quantitative determination of V&V intensity, the probability of a residual error in the simulation output, and the maximum acceptable residual risk, the opportunities for the application of V&V activities are explored in both breadth and depth in the following. However, it needs to be highlighted that all planed

and conducted V&V activities are driven by the identified worst case impacts, which result in an (abstract) risk statement. For the later graphical illustration of the dependencies between risk classification, model development, credibility building, and V&V, the abstract (outlined) representation of the risk classification process shown in Figure 7 is chosen, which stresses the need for the identification of the worst case impacts and the formulation of a risk statement.

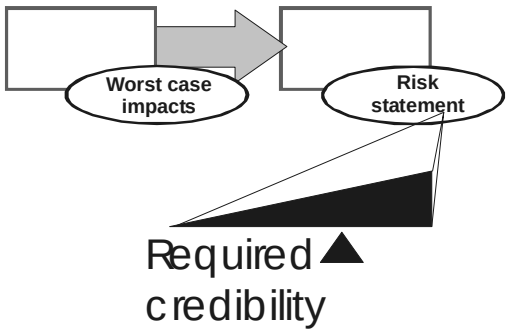


Figure 7: An abstract risk classification process

4.2 Real System Knowledge

The extend to which *information of the real system* is available (including theoretical and practical knowledge, reasonable assumptions, and "hard" measured data) varies with the observability of the real system. For several reasons the information about the real system may be incomplete, among them:

- The real system does not (yet) exist.
- The real system is under development and only partially exists.
- The real system is not exploitable or simply too complex.
- It is too dangerous or expensive to modify or manipulate the real system, its input, or its internal state for examination.

This problem has already been addressed by [Klir 1985] who defined *Levels of System Knowledge*. His concept was reused in [Zeigler, Praehofer, and Kim 2000], as shown in Table 2.

Level	Name	What we know at this level
0	Source	What variables to measure and how to observe them
1	Data	Data collected from a source system
2	Generative	Means to generate data in a data system
3	Structure	Components at a lower level coupled together to form a generative system

Table 2: Levels of system knowledge

At level 0, the border of the real system and its attributes and state variables are identified. At level 1, measured data is actually available, but the behavior of the system is still incomprehensive. At level 2, a symbolic behavior description can be provided, which allows to reproduce the data generated by the real system, which is still a black box. Insight into the structure of the system is gained at level 3.
Especially during validation, the model or simulation results are judged giving consideration to the knowledge of reality. The less is known about the real system, the less confidence in the suitability of the model can be established by comparing the symbolic model or simulation

results to the real system. The less empirically measured quantitative data is available (which for example is the case for large scale combat simulations [Davis 1992]), the more important become qualitative behavior descriptions and knowledge of the structure of the real system. For this reason, Klir's concept of Levels of System Knowledge is refined in the following by the distinction between the knowledge of the *structure* of the real system and the knowledge of its *behavior*.

4.2.1 System Structure

A real system can be considered as a black box with input and output parameters, without any structural information. This situation corresponds to Klir's Level 0 of System Knowledge, if no behavior data (data about the system dynamics) is available, otherwise to Level 1. However, the impossibility to directly analyze and formulate functional dependencies between the input and the output of the complete real system is often among the reasons for modeling complex systems. As indicated in section 1.1, one is then interested in the decomposition of the real system into a system of (sub-) systems whose functionality finally can be analyzed and formulated (top down), or in its construction by integration of subsystems with well known or analyzable behavior (bottom up). The identified subsystems and their hierarchical and peer-to-peer-dependencies are called *structure of the real system* in the following. The possible depth of insight into the structure of the real system depends on

- whether it is possible to identify components of the real system and their structural dependencies (Level of System Knowledge 3) and
- whether it is possible to actually manipulate or physically decompose the real system and to repeat the steps mentioned above iteratively.

If the real system is not or only partially observable or modifiable, the above knowledge may be gained or substituted by

- analysis of other symbolic or physical models of the real system (e.g., construction plans, prototypes), which were constructed for another purpose (e.g., construction), and may be completely, partially or not at all available,
- analysis of a related real system, which reflects the desired properties of the real system to a certain degree, or
- a postulate, which should be based on sound scientific assumptions.

At best, a complete multi-layered subsystems hierarchy can be created from direct detailed analysis of the real system and all of its subsystems. At worst, the real system cannot be decomposed at all and remains a black box.
The knowledge about the structure of the real system limits the V&V of the model or simulation results. As it will be shown in section 4.3, the decomposition of the model and its component-wise V&V against the real system is a powerful approach to managing V&V of complex models. The less is known about the internal structure of the real system, the more complex become V&V of the model.

4.2.2 System Behavior

When trying to understand the behavior of a real system in operation over time, one first decides, which aspects of the behavior might be of interest, and then chooses quantitatively measurable metrics to be able to measure the desired properties of the real system. (This corresponds to the Level 0 of System Knowledge; example of metrics for "object movement": velocity [m/s], acceleration [m/s^2], direction [rad].) This might be not feasible, if the available theoretic models of the real system are not sufficiently mature to provide such measures (example of metrics for "Human Behavior Representation": tiredness, concentration, creativity [Dompke 1999]). In this case, qualitative or ranked metrics accepted in the application do-

main should be used, and only in the worst case new metrics should be created without rigor scientific research. In addition, even if precise metrics can be identified, measurement errors must be taken into account, and statistical relevance of the measured behavior data should be ensured whenever possible (Level 1 of System Knowledge). The depth of achievable insight into the behavior of the real system (which again can be a subsystem of a higher level system) depends on the observability and controllability of the identified structural elements, i.e. the input and output parameters (I/O) of the real system and all its identified subsystems. The following cases can be distinguished (in extension to those defined in [Kleijnen 1999]):

- *Observation of neither input nor output*: No insight can be gained from observation of the system, only assumptions about its behavior can be made.
- *Output observation and unknown input*: The behavior of the real system can be observed, but one is not capable of identifying or measuring all relevant influences on the behavior of the real system. Maybe behavior patterns in the real system output can be identified. The system seems to be self-driven.
- *Unknown output, but input observation*: Only assumptions can be made how the real system reacts on the observable inputs, the system seems to absorb all external influences as a black hole.
- *Output and input observation*: Observing the input-output behavior of the real system, without controlling the input to the real system. Assumed cause-effect-dependencies between input and output can be formulated.
- *Output observation and control of input*: Observing the input-output behavior of the real system, while manipulating the input to the real system and/or its subsystems. This allows to create significant test cases for the real system and to perform aim-oriented examination. Assumptions about the behavior of the system can be confirmed or disproved. Unfortunately this is often not or only in limited form (lab conditions) possible, due to the same reasons of building the model (see chapter 1.2).

In analogy to structural analysis, the information gained by observation or manipulation may be augmented by information gained from observation or manipulation of related systems or other models. (However, then the questions of their suitability still remain.) In the best case, quantitative metrics exist for each subsystem, which can be exhaustively measured with only negligible measurement error. In the worst case not even a limited number of qualitatively defined high-level system outputs can be observed.
The non-availability of “hard” quantitatively measured data leads to less objective validation of a model or simulation results. For meaningful validation there should be at least input and output data for major subsystems available, otherwise the success of any validation attempt must be doubted. This evaluation is also shared by Sargant [Sargent 1999].

4.3 Model Knowledge

The maximum possible intensity of V&V is also limited by the availability of information about the model. Due to, e.g., protection of intellectual property or simply failure in documentation, the *available information about the model* itself may be limited. If – for any reason – necessary information cannot be obtained or reconstructed, V&V will be less intensive, as, if the information was available. In this situation the importance of quality assurance (quasi-) standards like the SW-CMM (section 3.2.1) or ISO9000 (section 3.2.2) becomes obvious: A developer who reached a high SW-CMM level or is certified for ISO9000 is more likely to produce adequate documentation than another, non-certified developer [Conwell, Enright, and Stutzman 2000]. However, the existence of documentation does not automatically imply availability for V&V. The age of the model also influences the likelihood of information availability: In the defense community, legacy models often come along without any mean-

ingful documentation of their Conceptual Model, used algorithms, or constraints and limitations.
Type and amount of *available model documentation* heavily influence the degree of credibility that the model can achieve, as the case studies [Gabor, Dietz, and Grahn 2000] and [Scholten 1998] show. In addition, the type of information available influences the selection of V&V techniques and the determination of achievable V&V objectives. Ideally, for the purpose of V&V, the available model information supports both a rigorous judgment of whether the model meets the acceptability criteria specified by the Structured Problem Description, as well as direct comparison to the knowledge of the real system, e.g., recorded real system I/O data (Klir's Level 1 of System Knowledge). To manage information about the model, in the following a *framework for the classification of model information* is introduced.

4.3.1 The Three-dimensional Model Information Space

Figure 8 depicts a three-dimensional model information space, in which it is possible to differentiate model information according to the information sub-space it belongs to. Later this classification supports the identification of required information for the achievement of a particular V&V objective, as discussed in section 4.7.4. The creation of this three-dimensional model information space is based on the introduction to M&S given in the sections 1.2 and 1.3.

- It is strictly distinguished between *information concerning conceptual aspects of a model* (which are most easily accessible using the Conceptual Model), *formal-mathematical aspects* (which are most easily accessible using the Formal Model), and *technical aspects* (which can be analyzed using the Executable Model).
- The *submodels are integrated hierarchically*, under the assumption that each model is explicitly or implicitly composed of submodels that depend on and interact with each other (see also section 1.1).
- It is distinguished between *the symbolic description of structure and behavior* (symbolic model) and the *interpreted behavior*.

These dimensions span an model information space. Examples for information sub-spaces in the three-dimensional model information space are:

- Subspace "interpreted behavior of the overall Executable Model" (which is close to the origin of the information space depicted in Figure 8), or
- Subspace "symbolic description of the atomic conceptual submodels" (which is the "most distant" cube in the three-dimensional model information space).

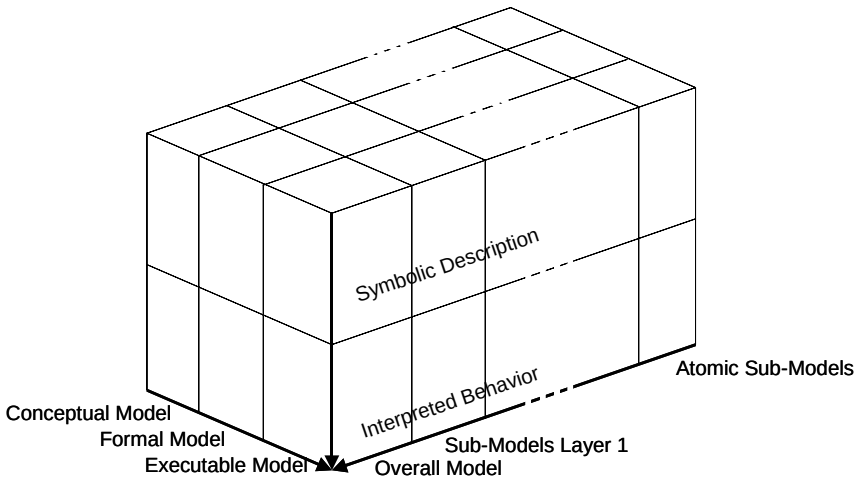


Figure 8: The three-dimensional model information space

Available information can be classified according to the sub-spaces in the information space. Examples:

- Input and output data recorded during an execution of the Executable Model belong to the information subspace “interpreted behavior of the overall Executable Model”. If data logs of the (main) submodels at the first decomposition level are available, these belong to the “interpreted behavior of executable submodel layer 1” (e.g., recorded I/O behavior of the federates in an HLA federation [IEEE 1516 2001]).
- If one mentally executes the conceptual description of the behavior of an atomic submodel (object) within the model (which is considered to belong to the sub-space “symbolic description of the atomic Conceptual Model”), one creates a mental image of the submodel’s behavior, which is classified as “interpreted behavior of the atomic conceptual submodel”.

The availability of information about a model depends on the stage of model development and the conditions under which the model is was developed. If no documentation requirements are formulated, the quality of information solely depends on the quality management process of the developer.
The information subspaces can be directly associated with the knowledge of structure and behavior of the real system. All structure knowledge of the real system may be used for the evaluation of the submodel hierarchy documented in the corresponding subspaces of the symbolic description of the model structure. All functional dependencies within the real system may be compared to the symbolic descriptions of model behavior. All data recorded at the real system may be used for comparison with the interpreted behavior of the model.

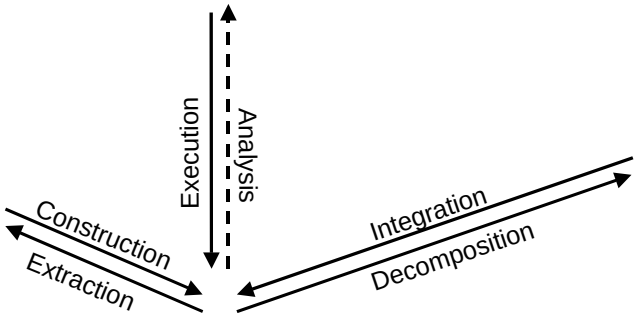


Figure 9: Activities along the axes of the three-dimensional model information space

If information about the model desired for V&V is missing, the dependencies between the information subspaces along the axes depicted Figure 9 may be used to create additional information about the model.

- *Construction*: The model is further developed, refined, and specified for a more advanced developmental stage. (This corresponds to the phases system analysis, formalization, and implementation of the model development process introduced in section 1.4.6).
- *Extraction*: Implicitly contained information is extracted. For example, if the description of the overall Conceptual Model is not available, one needs to disassemble and then to reverse-engineer the program code to extract the formal-mathematical aspects of the overall model description. These subsequently are interpreted to understand the underlying concepts.
- *Integration*: The submodels are assembled to create submodels of the next higher layer of the composition hierarchy. This is typically done during model development in the phases system analysis, formalization, and implementation. Also the recorded I/O of the submodels can be integrated to yield the I/O of the associated next higher layer submodel.

- *Decomposition*: A submodel is decomposed into submodels on the next lower layer of the submodel hierarchy. This usually is performed for separate examination of each individual submodel.
- *Execution*: Behavior data (model output) is generated by interpretation of the symbolic description of structure and behavior based on model input. This typically is done during the experimentation phase, when one uses the executable binary file on a hardware platform to generate simulation results. Other forms of execution are mental or symbolic execution of the conceptual (system analysis) or formal (sub-) models (formalization). During the implementation phase, the model is executed for the purpose of error detection (Testing).
- *Analysis*: Formulation of the symbolic structure and behavior description of the model is based solely on model input and output. This is as difficult as the analysis of the real system, and stated here only for reasons of completeness.

The availability of information in the subspaces heavily influences the possible choice of V&V activities. For example, the examination of the symbolic description of the Executable Model for the purpose of model validation usually is expensive (code analysis, more suitable for verification), thus, if possible, here examination of the interpreted behavior is more efficient (testing). The evaluation of the behavior of the Conceptual Model is not very reliable (interpretation by mental execution, often only qualitatively), the examination of the symbolic description is more suitable (analysis).
In general one is interested in validating the behavior of the overall Executable Model. Because this often is not directly possible, one must find a path through the information subspaces that allows the justified conclusion that the originally intended goal has been achieved. How the available information influences the V&V of a model is discussed in section 4.6. The concepts of the three-dimensional model information space have been previously published in [Brade, Maguire, and Lotz 2002] and [Kilikauskas et. al 2002].
The structure of the submodel hierarchy may vary through the different intermediate products (conceptual, formal, Executable Model). Even monolithic legacy models have been built upon Conceptual Models designed in a hierarchical, modular way. Easy projection from the conceptual submodel hierarchy over the mathematical-formal submodel hierarchy to the executable submodel hierarchy is desirable, but not necessary.

4.3.2 Populating the Three-Dimensional Model Information Space

In most cases the reconstruction of missing information is extremely expensive (especially for analysis and extraction). As a consequence, the minimum requirement for a model development process to support successful application of V&V activities is to *provide intermediate products that populate the three dimensional information space*. Otherwise the operations described above need to be applied during V&V to (re-) construct required missing information. The model development and execution process introduced in section 1.4.6 meets this minimum requirement. For its below integration into a processes framework, its graphical representation was adapted as shown in Figure 10. The arrow shows the general direction of model development, the black boxes the main developmental phases, and the ellipsoids represent well-defined intermediate products created during each phase. Iterations due to the change of requirements or additional knowledge gain during modeling are not explicitly graphically visualized, but do not interfere with the chosen representation. The expected activities in the development phases and their associated intermediate products are specified in detail in Appendix A, thus a solid representation is chosen for them.

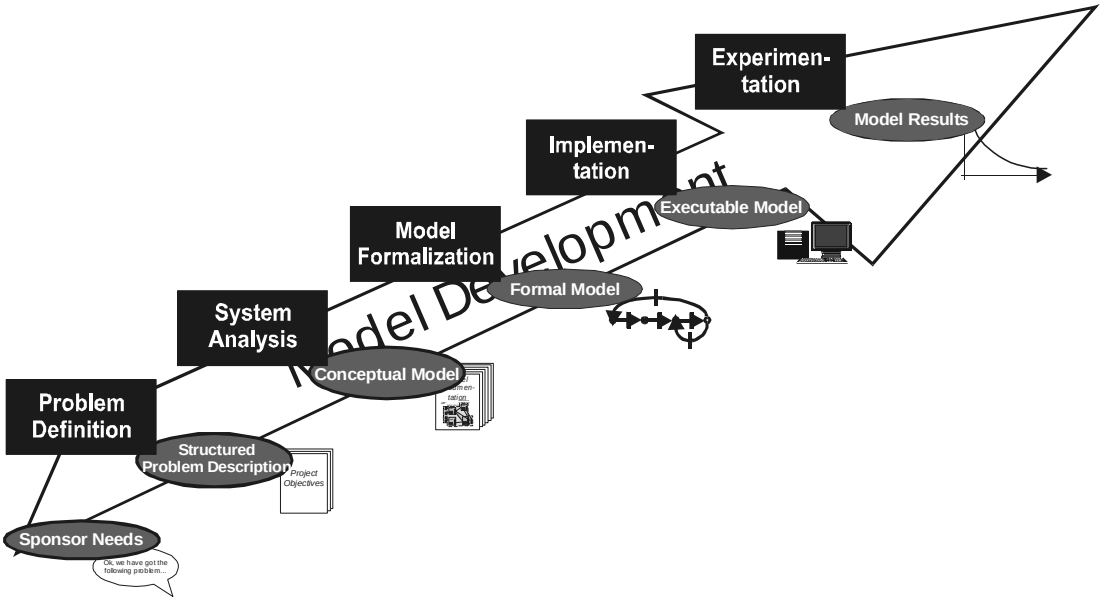


Figure 10: The model development and execution process of section 1.4.6 represented alternatively

4.4 Establishing Credibility

To establish the credibility of a model or simulation results means to increase its perceived correctness and perceived suitability, and therefore to make plausible that no relevant errors or inadequacies remain in the model or the simulation results. This section motivates the introduction of the vague term *credibility* that addresses "the quality or power of inspiring belief" [Merriam Webster 2003].

According to the principle stated in section 3.1.6, in most cases there is no proof of correctness and suitability of a non-trivial model or its simulation results. As a consequence, one never definitely knows when to terminate V&V activities, unless a sufficient number of errors and inadequacies was found to render the model or simulation results unsuitable or incorrect (*failure of V&V* or *successful falsification*). This suggests that the perceived suitability and perceived correctness can only inspire the believe that a model or simulation results actually are suitable for an intended purpose and correct. Based on the power or quality of the inspiring belief, which is founded on the intensity and the rigor of the conducted V&V activities and their successful communication, a final decision is made, whether the model or simulation results can be used or not – always with a residual risk left.

Following the principles of V&V, the credibility of one specific model or set of simulation results has to be examined explicitly, as the credibility indicated by the credibility of the developer (based on its SW-CMM level, an ISO9000 certification, or the developer's reputation) may be to high (*model users risk*) or to low (*model builders risk*).

In order to apply simulation results, the demonstration of the correctness and suitability of the model is required, but not sufficient. The credibility of simulation results is based on three main influences:

(1) The credibility of the *model* used for experimentation and its embedded data,
(2) the credibility of the *runtime input data*, and
(3) the credibility of the ways of *operation* or application of the model.

If only one of the three items is not credible, the correctness and suitability of the simulation results must be doubted. Figure 11 visualizes this concept.

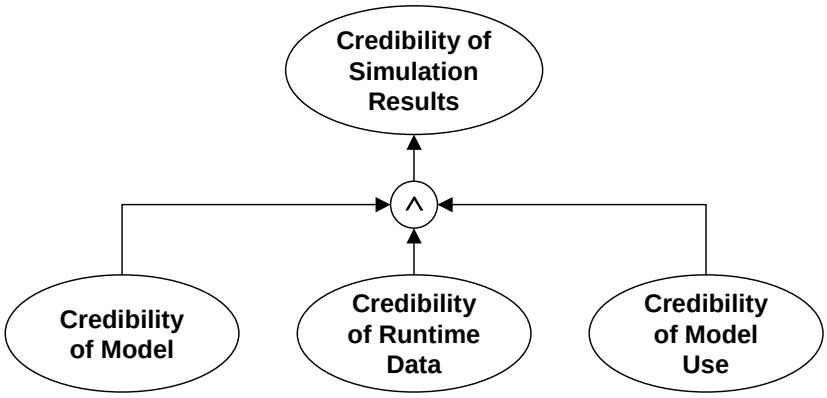


Figure 11: Credibility of simulation results

4.4.1 Credibility of the Model

The credibility of a model is based on its perceived suitability and perceived correctness, always giving consideration to the intended purpose. To establish credibility of a model, the three-dimensional model information space and the dependencies between its sub-spaces introduced in section 4.3.1 are used.

- *Credibility of the Conceptual, Formal, and Executable Model*: There is a strong dependency between the credibility of the Conceptual Model, the Formal Model, and the Executable Model. If the Conceptual Model is credible with respect to the knowledge available about the real system, here it is assumed that the contents expressed by the formulas of the Formal Model are also credible, if it is shown that it correctly (with respect to the modeling formalism) and suitably reflects the Conceptual Model (traceability). The same is true for the Executable Model: If the formulas of the underlying Formal Model are credible, and it is shown that they are suitably and correctly encoded, the Executable Model becomes credible. Thus, to establish credibility of the Executable Model incrementally, one starts with validating the Conceptual Model against real system knowledge (giving consideration to the intended purpose), and then reduce further work to verifying internal correctness of the Formal Model and complete consistency with the Conceptual Model, and do the same for the Executable Model. Incremental credibility building should be conducted in parallel to the advance in model development over the phases system analysis, formalization, and implementation.
- *Credibility of the submodel hierarchy*: A model is considered to be credible, if all its submodels are credible and interact in a credible way. The bottom-up approach along the "integration" axis (see Figure 9), is to "integrate" credibly interacting credible submodels to credible submodels on the next hierarchical level. If it is not possible to directly establish credibility of the overall model, it needs to be "decomposed" top-down into credibly interacting submodels, until it is feasible to show the credibility of the submodels. This should be done during each of the phases system analysis, formalization, and implementation.
- *Credibility of symbolic description and interpreted behavior*: If the chances are limited to directly observe behavior of both the complex model and the real system, the extrapolation of expectable behavior from the symbolic structure and behavior description is desirable to increase credibility of the interpreted behavior (and vice versa). For early error detection during model development it is important to ensure that the conceptual structure and behavior description is credible, which finally will be confirmed, when the observed behavior of the Executable Model is credible. Having successfully established credibility of the symbolic model, "execution" is expected to

yield credible observable behavior (and vice versa). This should be done during system analysis and implementation, but also be considered during formalization.

To achieve an increase of credibility of the behavior of the overall Executable Model, one can approach this challenge by creating paths through the three-dimensional model information space in Figure 8. Under the assumption that the sponsor/beneficiary is exclusively interested in the quantitative model results, the end point of these paths is always the origin of the three-dimensional model information space; the starting points vary with the information available.
For example: Starting with establishing credibility of the structure and behavior description of the conceptual atomic submodels by evaluating the differences between them and the associated real subsystems. One may continue with establishing credibility of the structure and behavior descriptions of the formal atomic submodels by showing their correctness and complete consistency with their conceptual descriptions. After subsequent successful demonstration of the consistency between the Formal Model of the atomic elements and their implementation, one may proceed establishing the credibility of the behavior of the executable atomic submodels. Integrating them bottom-up layer by layer and building credibility in each layer of executable submodels, most probably results in a credible overall model. (Actually, this is a generalized description of the CLIMB approach [Flight Mechanics Panel Working Group 12 1985], as discussed in section 3.2.6, which can be considered as a special path through the three-dimensional model information space).
As a consequence, the V&V process to be developed should support *incremental credibility building*. This means, the credibility of information of a particular information sub-space may be both directly increased by comparison to external specifications, data, or knowledge, or build upon credible information from another information sub-space by ensuring consistency and completeness with the information in this sub-space.
To achieve the aims of this thesis, an abstract representation of the credibility building process is adequate (outlined representation). The graphical illustration of the credibility building process shown in Figure 12 stresses the concepts that

- credibility is built stepwise in parallel to model development and
- evidence (modular V&V results) on which credibility is build needs to be communicated.

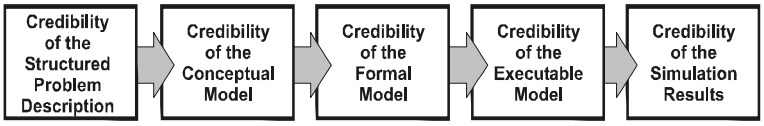


Figure 12: Abstract representation of the phase-wise credibility building process

4.4.2 Credibility of Runtime Input Data and Embedded Data

When establishing credibility of data, different categories of data must be distinguished:

- *Raw data*, which is directly available in a data base. Raw data is created by measuring and recording at a real system, or is generated by other means, e.g., experimentation with another model.
- *Aggregated data*, which is directly available in a data base. It is created by modification of raw data.
- *Parameterized data generators*, which create data according to a given distribution function and their initialization data.

The credible use of raw data requires the assessment of the following characteristics:

- *Age*: Data for simulation must be up to date. Data that does not reflect the current properties of the real system usually is not credible.
- *Real system of origin*: Data shall be taken from a suitable real system. This is relevant, if (for any reason) it is not possible to gain data from the real system, which is simulated and the data is "transferred" to the simulated system.
- *Measurement method*: It should be credible that data is measured precisely enough for the intended purpose of the model. That includes that the sampling rate is sufficiently high, the rounding errors are insignificant, and the inaccuracy of the measuring device is irrelevant.
- *Interval*: The data *interval* should be suitable as model input (valid input interval or domain) and suitable for the achievement of the goals of model operation.
- *Internal consistency*: Data is not credible, if it is shown to be internally inconsistent.

The following technical and interoperability issues should be addressed:

- *Data format*: Data should not be used for simulation, if it is not credible that there will be no technical complications when accessing the data base.
- *Compliance with data models*: For substantive interoperability of models in distributed simulations usually a common data model is required. This can be achieved by compliance with a standardized data model.

Often models generate their own input data (*self driven simulation*). For this purpose, usually a generator for a parameterized (stochastic) distribution function is used, randomly generating the desired input data. For the judgment of their credibility the following should be considered:

- *Goodness of fit*: The empirical distribution of the generated data must fit the empirical distribution of the chosen reference data, otherwise it is no suitable substitute for the real data. Both the parameter values and the distribution function may have to be adjusted.
- *Independent and identical distribution*: To allow the substitution of real data by stochastically distributed data, the assumption that the real data is independently and identically distributed (IID) must be justified. If a non-stationary process is modeled (time-varying arrival rates), distribution function or parameters must be adjusted [Vincent 1998].

A special class of runtime data in simulations are the *random numbers*. They usually are created during a simulation execution as a stream by a *random number generator*. The structure of the random number generator already provides insight into the credibility of the randomness of the generated numbers. [L'Ecuyer 1998; Keppler 1993] discuss several approaches to random number generation, initialization (*seed*), their advantages, and their shortcomings.

- *Correlation*: The degree of correlation in a set of numbers must be low enough to credibly make it a set of "random numbers".
- *Period length*: Among the most important aspects of a random number generator is the period length of the generated numbers – on modern computers a period length of 2^{16} may be used up during a few minutes (of wall clock time), disqualifying the generator as a baby toy.

As a consequence for the V&V process, data acquisition, data measurement, data transformation, and data use shall be considered during its phases, especially during the V&V of system analysis and experimentation.

4.4.3 Credibility of Model Operation

The best model may become dangerous, if applied in the wrong manner [Pawlikowski, Jeong, and Lee 2002]. [Kleijnen 1998] defines the design of experiments (DOE) as "selecting the combinations of factor levels that will automatically be simulated in an experiment with the simulation model" as sub-discipline within mathematical statistics. Even if the experiment design is not based on a mathematical foundation, the following influences on the credibility of model operation need to be considered:

- *Design of experiments*: It has to be credible that the design of experiments (harmonized set of simulation runs) allows the drawing of conclusions. Especially for stochastic simulations the design of experiments is a non-trivial problem.
- *Runtime environment*: The runtime environment of a model may disturb the simulation run. Faults in the runtime communication structure, random number generators that create highly dependent "random" numbers, and other malicious effects may result in non-credible model operation.
- *Completeness*: To achieve the goals of simulation, scenarios are most often predefined in the requirements. It shall be credible that all required scenarios have been simulated.

This implies that during the V&V of experimentation adequate model operation has to be ensured.

4.5 Remarks on V&V Techniques

Numerous techniques that may be used for V&V of models and simulation results are available from different research and development areas, including *requirements engineering*, *system analysis in the application domain*, *M&S theory*, *software engineering*, *experiment design in the application domain*, and *experiment design in computer-based simulation*:

- *Requirements engineering* addresses the communication problem between the user and the developer and its solution, and offers methods to express "what you want" [Partsch 1991]. The use of these methods makes it more likely that these requirements precisely direct the development of the final product ("what you get"). Although the development of requirements for a model and simulation slightly deviates from software requirements engineering, numerous relevant aspects can be transferred to support the achievement of a complete, consistent, correct, suitable, unambiguous, and testable Structured Problem Description. To formalize the requirements, and to express logical dependencies formally, mathematical logics are suitable.
- No general statement can be made about the applicability of techniques from the application domain (including *system analysis techniques*), because approaches for model validation vary according to the tasks within the countless application domains. However, for quantitative models there is always the desire to validate them against empirical data. For the V&V of simulation results, statistical methods (data aggregation and statistical tests) play an important role [Kleijnen 1998].
- Especially *error detection techniques* and *verification techniques* are available for different modeling formalisms, which originate from M&S theory, including, e.g., syntax and semantic checkers, or deadlock detection methods (aliveness) and reachability graphs for state transition models. Numerous errors within the Formal Model can be detected following approaches of M&S theory [Zeigler, Praehofer, and Kim 2000; Hiller and Liebermann 1997].
- In the area of *Software Engineering*, numerous *test and analysis techniques* are known that support the detection and elimination of errors in the program code [Myers 1979; Liggesmeyer, Sneed, and Spillner 1992; Beizer 1990; DeMillo et al. 1987]. These

techniques are highly recommended for the examination of the program code that implements a model (Executable Model), not only for the purpose of software quality assurance, but also for V&V of the model. Although M&S are significantly more than just encoding a piece of software, with these software analysis and test techniques a variety of techniques exists that can be used after slight modification to support assessment of the correctness and suitability of a model or simulation results.

Popular summaries presenting unmodified techniques from the above areas as V&V techniques are given in [Defense Modeling and Simulation Office 1996 and 2000; Balci 1990]. These summaries can be considered as a starting point for subsequent work, as in general it is not possible to directly apply these techniques for V&V of the intermediate product without modification. The Structured Problem Description, Conceptual Model, Formal Model, Executable Model, and Simulation Results have much in common with the products created during a structured software development process, but there also are non-negligible differences. To allow the efficient and effective execution of V&V techniques, they need to be adapted to each intermediate product individually. In chapter 5, conducted adaptation and harmonization of V&V techniques is documented.

4.5.1 Demarcation of Validation, Verification, Testing, and Analysis

For most of the V&V techniques that are summarized as such in literature [Defense Modeling and Simulation Office 1996 and 2000; Balci 1990; Shannon 1975] it does not seem to be reasonable to distinguish them as techniques for either model verification or model validation. Most of them actually are analysis or test techniques that can be used for *both* validation and verification – it depends on whether the examination or evaluation of the intermediate product focuses on its compliance with well-defined rules, or whether it supports the assessment of suitability for the intended purpose with respect to the real system. More often than not, the use of a V&V technique on an intermediate product yields a *V&V result* or *product* (V&V evidence), which contains extracted, previously hidden information, that subsequently is evaluated according to precisely specified rules (that may be available as a standard, guideline, or other unambiguous specification) *or* for suitability with respect to the intended purpose and the real system, respectively.

For software quality assurance, one usually distinguishes between *analysis*, *testing*, and *verification*. Analysis and testing can both be performed for the purpose of V&V of models and simulation results. The definition of software verification is stronger than the definition of model verification, as software verification usually requires the *proof* of correctness (not only its demonstration).

Under consideration of the taxonomy used for software testing, analysis, and verification in [Liggesmeyer, Sneed, and Spillner 1992], and according to the three-dimensional model information space introduced in section 4.3, for the V&V of models and simulation results, in the following it is distinguished between

- techniques for *analysis of the symbolic model and its submodels* in all stages of model development (section 4.5.1.1),
- techniques for *analyzing input, embedded, and output data* (section 4.5.1.2), and
- techniques that support the *development and evaluation of high yield test cases* (i.e., test cases that are most likely to reveal existing errors [Myers 1979], section 4.5.1.3).

4.5.1.1 Static Model Analysis

For static model analysis the (symbolic) description of the problem and the (conceptual, formal, executable) model are subjected to examination for correctness and suitability. The term “Static Model Analysis” is chosen in analogy to the terminology used for software analysis

[Liggesmeyer, Sneed, and Spillner 1992], where static analysis (of software) is defined to "include code verification and numerous analytic measures that provide additional information about the code". The general purpose of static model analysis is the aim-oriented extraction of particular model aspects that are (implicitly) hidden in the model description, and to subject these special aspects, now revealed, to further examination. The stage of development of the model, and the type of the intermediate product subjected to examination heavily influence the selection of an effective static analysis technique.
Methods for static model analysis always require insight into the symbolic model, and often also serve as preparation for "white box" and "gray box" testing techniques. In addition to directly revealing incorrectness or unsuitability of the analyzed intermediate product, analysis results can later be reused for further V&V activities, as, e.g., testing of the Executable Model. The gained knowledge of internal dependencies of the intermediate products positively influences the creation of test cases with a high yield.
As indicated with the three-dimensional model information space in section 4.3.1, hierarchical decomposability allows to handle symbolic models of high complexity. If the symbolic model is decomposable, the analysis should be conducted component- or partition-wise, decomposing it top-down, integrating it bottom-up, or meet-in-the-middle as appropriate.

4.5.1.2 Test Case Development

Another possibility to establish the credibility of a model is the sample survey examination of its behavior. When testing a model, its *expected behavior* should be anticipated, as it is recommended for software testing [Beizer 1990], and subsequently compared to the *observed behavior*.
If the structure of the model under test is only of secondary importance, in analogy to software testing this will be referred to as model "Black Box Testing" [Becker 1989]. The test setup can either be based on the modeling requirements or be purely randomly. Due to the size of the output space of non-trivial models, exhaustive analysis usually is not possible, and tests should be carefully designed to allow the investigation of the most interesting sub-spaces of the output space (*non-exhaustive testing*). In contrast to Black Box Testing, model "white box testing" is based on the knowledge of the internal structure of the model in all its representation forms (Conceptual Model, Formal Model, Executable Model) gained during static model analysis, and will be used to confirm or disprove expected behavior by aim-oriented stimulation of the Executable Model. The identification of the most interesting sub-spaces can be conducted following different approaches; several model analysis techniques yield useful results (e.g., cause-effect graphs or control-flow graphs) that support the search for high yield test cases.
Software testing usually is implemented as execution testing, field testing, functional testing, or product testing [Defense Modeling and Simulation Office 1996 and 2000]. Depending on how close the (nearly finished) product is to its release date, the audience involved in testing changes, the number of testers increases, and tests become more in number, but less focused and more use-oriented. (Alpha Testing, Beta Testing, Acceptance Testing). In addition to the "normal" model output, supplementary data can be recorded by instrumentation of the model.
According to the three-dimensional model information space, these methods support the examination of the correctness and suitability of the I/O behavior of the overall model and all of its submodels (if separately available). If partition testing or submodel testing is possible, i.e., separate submodels can be executed and their behavior can be observed, either bottom-up or top-down, as suitable.

4.5.1.3 Data Analysis

Under the assumption that with program crashes and obviously erroneous model output is dealt during software quality assurance, the difference between correct and incorrect, or suit-

able and unsuitable model behavior is very subtle. The detection of incorrectness or unsuitability requires rigorous analysis of the outcome of each test case. For the analysis of data, regardless of whether it is input, embedded, or output data, it must be distinguished between "soft" qualitative information and "hard" quantitative data. Techniques based on statistics and mathematical logics are intended to be used for the analysis of quantitative (or at least ranked, i.e. ordered) data. Depending on the aim of data analysis, either the internal structure of the (sub-)models used to generate the data or its specification is of importance, or comparison data is required. The reliability of the comparison data depends on the maturity of research, measurement methods, statistical relevance, and measurement tools in the particular field (see also section 4.1 on real system knowledge).
After generation, model results can be directly examined, or they can be aggregated to a higher level, which allows to deal with greater amounts of data by, e.g., fitting to a distribution function, or calculation of the mean value. When aggregating simulation results created by stochastic models, it must be ensured that enough "independent" simulation runs have been executed to guarantee the statistical significance of the simulation results.
Data analysis techniques often yield results that again need to be interpreted by an expert to be useful for further model improvement. Therefore it seems to be advisable to use expert opinion supported by formally defined techniques for the examination of the correctness and suitability of simulation results. Statistical tests can be applied to test a hypothesis formulated by an expert and to create confidence in this statement, based on the extents of the samples examined. Expected model behavior may be specified using a suitable formalism, e.g., temporal logic (see section 5.3). This specification can then be used to check data sets for violation (even during runtime), or for automatically post-processing the simulation output.

4.5.2 Expert Opinion as Validation Evidence

With often no mature quantitatively defined metrics for validation available, suitability needs to be judged by an instance that knows the meaning of the terms "residual risk" and "responsibility". Most V&V techniques yield V&V results that support a more objective judgment of the credibility of the model or simulation results. But even if a proof of correctness has been successfully completed, it still must be judged, whether the specification used for the proof is suitable. The unavailability of truly objective V&V techniques and measures leads to the distinction between "highly subjective" V&V techniques (which are mainly based on unfiltered expert opinion), and "nearly objective" V&V techniques (with a mature formal foundation).
Expert opinion is used for direct or indirect judgment of model credibility. On the one hand, the expert can be directly confronted with the model, and asked to state a (highly subjective) opinion. The main disadvantage of direct judgment is that it is rarely repeatable, as soon as another expert gets involved. The "inspiring power" of this type of evidence (documented expert opinion) solemnly depends on the expert's reputation. Alternatively, the expert can evaluate knowledge concerning particular model aspects explicitly excerpted from the model, while all other model aspects are hidden, and thereby judge the credibility indirectly (nearly objectively) without distraction. In this case, next to the expert statement there is the documentation of the extracted model aspects as evidence. As to err is human, and according to the argumentation in section 3.1.4, it is desirable to increase the share of nearly objective V&V techniques whenever possible, and thereby to enhance repeatability of V&V activities. However, in this case it must be ensured that these activities explicitly cover all relevant model aspects.

4.6 The Dependencies Visualized and Roles in V&V

Following the introduction of risk, system knowledge, model knowledge, credibility, and V&V techniques in the context of the thesis, these influences on V&V are brought together. To visualize the dependencies between risk classification, model development and execution,

V&V objectives identification, V&V results creation, and credibility building, Figure 13 was drawn. In addition to the previously introduced processes, the V&V process, explained in section 4.7, with phases (columns) and V&V sub-phases (gray boxes 1.1 – 5.6) is depicted below. The intermediate products from model development serve as input to the V&V process. The risk classification process drives the selection of V&V sub-phases, which will be processed during V&V execution. The credibility of the model and its simulation results is based on V&V results (evidence) associated with the individual V&V sub-phases. The form of the V&V process already indicates the numerous feedback loops that allow to increase credibility during model development iteratively. The V&V process, which will be referred to as *V&V Triangle* in the following (due to its graphical representation) is specified in Appendix A.
To plan and implement the V&V activities, V&V roles can be identified, in analogy to the roles in the M&S development process:

- *V&V Agent*: This role manages all V&V activities that shall be conducted. The actor must be knowledgeable in V&V methodology and tools support, and is responsible for ensuring that the required degree of credibility is achieved or identified errors are reported to the user and/or the accreditation agent.
- *Subject Matter Expert*: The qualification requirements for this role are identical with those for the SME in M&S Development. He is responsible for ensuring the association between model and real system being suitable and correct.
- *Modeling Expert*: The qualification requirements for this role are identical with those for the Modeling Expert in M&S Development. He is responsible for ensuring the correctness and suitability of the Conceptual Model and the Formal Model.
- *Programming Expert*: The qualification requirements for this role are identical with those for the Programming Expert in M&S Development. He is responsible for ensuring the correctness and suitability of the implemented Executable Model.
- *Accreditation Agent*: This role represents the user who accepts of rejects the model or simulation results for the intended purpose. He must be knowledgeable in theory and practice of M&S and the application domain.

The generalized V&V process, which is specified in the following section is embedded into this framework.

4.7 Structured V&V – The V&V Triangle

The V&V Triangle is a generalized process for the stepwise V&V of models, their intermediate products, and simulation results. As there is no “one size fits all” solution, by providing a harmonized summary of V&V objectives it becomes a process model for mastering V&V of models and simulation results, and thereby serves as structured foundation for further V&V research. As a generalized process, it covers most of the approaches to V&V known today. It is depicted in Figure 14 and integrated into the framework introduced in section 4.5, which implies that

- its inputs from model development are well-defined intermediate products that are created during model development (as indicated by the model development process introduced in section 1.4.6) giving consideration to the chronological order of their availability;
- the intensity and rigor of V&V are variable, according to the risk incident to the intended purpose of model use, supporting incremental establishing of credibility (according to the concepts introduced in section 4.4);
- all V&V objectives identified within the process are harmonized with each other;

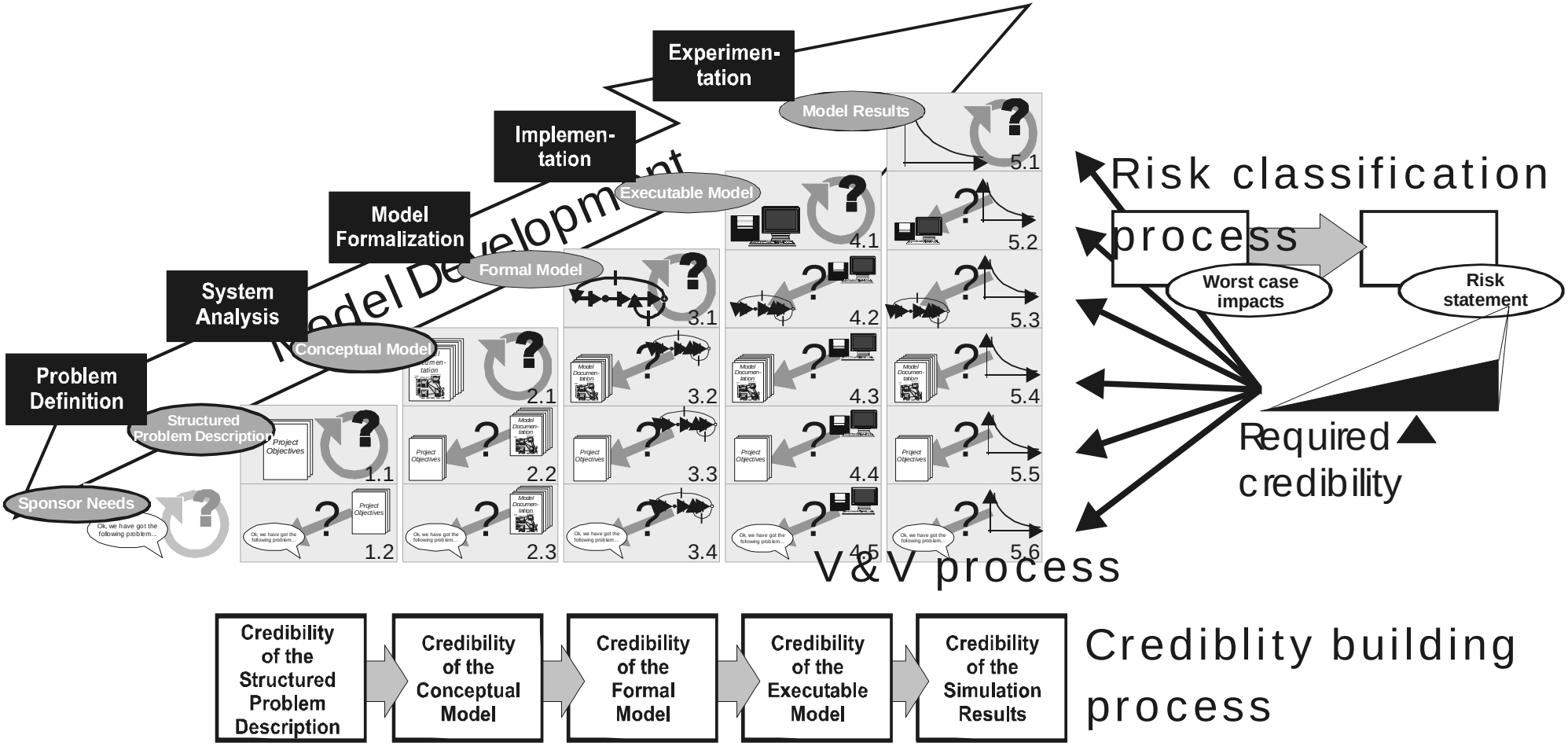


Figure 13: V&V process framework

- previously created V&V evidence is reused efficiently during subsequent V&V activities;
- its output is the modular documentation of the achieved V&V objectives, the conducted V&V activities, and the V&V results, which can be used to inspire the belief in the correctness and suitability according to the identified risk.

To achieve the goals of this thesis identified in section 2.2, the design of the V&V Triangle combines the fundamental approaches to V&V (introduced in section 3.1.6 giving consideration to the principles identified in section 3.1). It provides

- requirements concerning real system knowledge (according to the concepts introduced in section 4.1);
- detailed documentation requirements for the intermediate products (according to the three-dimensional model information space introduced in section 4.3);
- a summary of types of errors, which are most likely to be hidden in the intermediate product on which the V&V activities are focused;
- V&V main phases to assess the correctness and suitability of each particular intermediate product (see section 4.7.3);
- V&V sub-phases with generalized V&V objectives to fully explore the potentials of the examination of specific types of intermediate products and external information (see section 4.7.4);
- guidance on how to apply V&V techniques to achieve the V&V objectives, and on how to reuse previously created V&V products;
- an overview of the dependencies between different V&V objectives and the influence of repetitive examination on the credibility of the model and simulation results.

Within the V&V Triangle, three consistent views on V&V of a model and simulation results are combined:

- The errors view: All V&V activities are conducted to confirm the absence of errors in the intermediate products, or to reveal them. In Appendix A, each specification of an intermediate product is immediately followed by a summary of errors, which must be expected and drive most V&V activities.
- The objectives view: From the identified errors explicit V&V objectives are derived, which allow to focus all V&V activities. With errors propagating through the intermediate products, V&V objectives and their dependencies are defined for each V&V sub-phase.
- The techniques view: V&V Techniques are discussed, which help to achieve the proposed goals. They are summarized and associated to the objectives defined for each V&V sub-phase.

The V&V process is organized as a (lower) triangular 6×5 matrix, with

- the initial product *Sponsor Needs* and the five intermediate products *Structured Problem Description*, *Conceptual Model*, *Formal Model*, *Executable Model*, and *Simulation Results* (section 1.4.6) arranged along the horizontal dimension. (The five intermediate products are also arranged along the vertical dimension, as depicted in Figure 14.)
- the columns of the triangular matrix representing the *V&V main phases*, which are associated with the intermediate products placed (left) above them;
- intersections between the columns and rows (that are associated with the intermediate products to their upper left), which split the V&V main phases into *V&V sub-phases* (gray boxes in Figure 14).

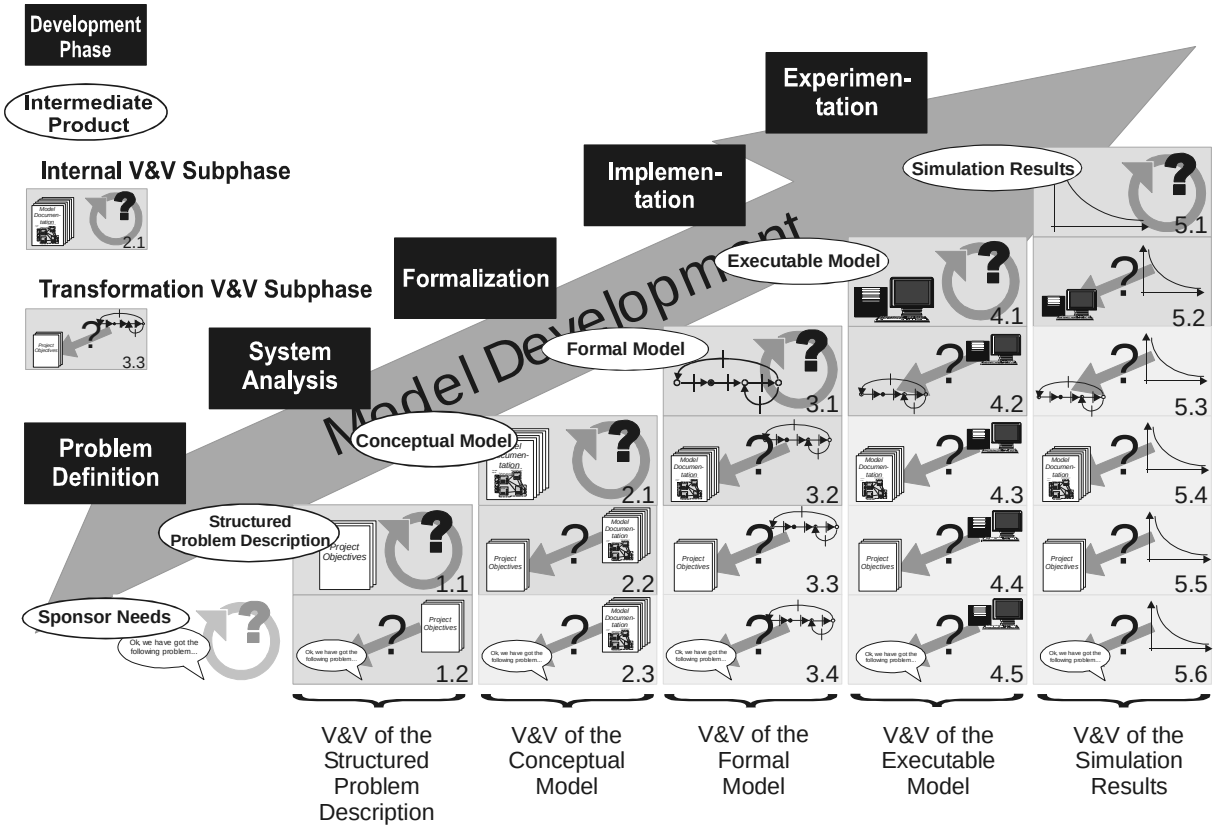


Figure 14: The V&V Triangle

It becomes obvious that the V&V process is embedded into the complete model development and execution process. With three different forms of model representation (conceptual, formal, executable), it focuses on model V&V, but also considers V&V of the Structured Problem Description, and addresses the need for correct and suitable Simulation Results. The first concepts of the V&V Triangle were published in [Brade 2000].

4.7.1 Intermediate Products

The V&V Triangle is not limited to a particular model development process, but product-oriented. The process-orientation aspect is only touched by the assumption of traceability through the intermediate products. The intermediate products may be available in different versions (as typical for a cyclic or spiral development process), which requires reasonable configuration management. The development phases depicted as black boxes in Figure 14 symbolize sets of activities that need to be executed at least once to transform one intermediate product into its successor. By clearly defining required contents of the intermediate products, the V&V Triangle provides a *guideline for documentation* of each step of model development from the V&V perspective. Whenever those (extensive) documentation requirements are satisfied, there is a solid foundation for both execution of V&V activities and for reuse of the knowledge documented in the intermediate products gained during the associated development phase.

4.7.2 Failures, Errors, and their Propagation

Among the aims of model improvement is the avoidance of any type of failure during model development or error within an intermediate product, or to detect them and initiate their correction as early as possible. When *errors* or *error classes* are known that have to be expected,

the process of error avoidance, search, and detection, and therefore the chance of successful model development can be increased by well directed V&V activities.
In literature there is a clear differentiation between *faults*, *errors*, and *failure*. In [Smith and Wood 1987] for example this differentiation is used to deal with the issue of fault tolerance in software. There a *fault* (e.g., a wrong variable is incremented in the program code) causes an erroneous state (*error* – e.g., in the main memory the value of the relevant variable is too low), leading to *failure* in providing the required service (e.g., a program function call returns a too low number of items in a storage hall). The failure of a subsystem may be considered as a fault in the next higher subsystem, resulting in chains of failures and errors. However, in this document these terms are used in a slightly different manner, as Smith and Wood focus on the *execution* aspect of software, whereas here, error propagation during the *construction* of the model is of main interest (compare with section 4.3.2). In the context of this document,

- whenever the developer fails to perform a developmental action or task correctly, this will be called *failure*, and as
- the failure most probably leads to incorrect or unsuitable information within the associated intermediate product, this yields an intermediate product with an *error*.

Or, in other words, *failure in development* causes *errors in the product*. (This is consistent with the usage of the term "error" in [Shannon 1975] or [Vengheluwe 2001].)
If one is interested in improving model development, *countermeasures to avoid failures* should be introduced: If the scope is on judging the correctness and suitability of a product, *methods and techniques for error detection* should be identified and used. [Shannon 1975] points out that there often are errors in design, errors in programming, errors in the data used, errors in the procedure of model use, and errors in interpretation. [Vengheluwe 2001] discusses modeling errors due to an improperly defined experimental frame, due to an improperly characterized model structure, and due to inaccurate estimates of model parameters. However, both stop at this level of description and give no further refinement.
Regarding intermediate products, two overall *error classes* are identified:

- *Internal errors* (exactly one intermediate product is analyzed): If one examines the Structured Problem Description, the Conceptual Model, the Formal Model, the Executable Model, and the Simulation Results independently from each other as self-contained products, internal inconsistencies, inadequacies, incompleteness, or incorrectness with respect to external rules, regulations, facts, assumptions about the real world, or comparison data may be detected. The intermediate product internal errors include errors introduced by faulty external information, which is won independently from previous model development and not documented in any preceding intermediate product. The expectable errors depend on the intermediate product subjected to examination.
- *Transformation errors* (Intermediate products are compared pairwise): The developmental advance of the model is evaluated under explicit consideration of a previously created intermediate product, on which the current product is based. If relevant contents of the preceding intermediate products are not completely and consistently reflected by the succeeding intermediate products, consistency and completeness requirements between the intermediate products have been violated. The type of the relevant contents depends on the intermediate product.

Practical experience and discussion with professional model developers revealed the error and failure types that occur during model development and intermediate product documentation summarized in Appendix A. The author does not claim these lists to be complete. These typically expectable failures and errors are classified and described, including

- requirements specification failures and errors in the Structured Problem Description,

- analysis, modeling, or projection failures and errors in the Conceptual Model,
- formalization failures and errors in the Formal Model,
- implementation failures, and errors in the Executable Model,
- use, experimentation, or operation failures and errors in the Simulation Results.

As already motivated in section 1.4.6, failures in configuration control are deliberately not included into this examination. Also failures that occur during interpretation of Simulation Results (in a waterfall process conducted subsequently to experimentation, see Figure 2) are out of scope of this thesis.

4.7.3 V&V Main Phases

The V&V Triangle visualizes the dependencies between the intermediate products and the V&V phases. One V&V main phase is assigned to each of the five intermediate products, which ideally are gained sequentially during model development.

- During V&V of the Structured Problem Description (*Structured Problem Description V&V*; column 1.x with x = 1,2 in Figure 14) shall be shown that the Structured Problem Description is suitable and correct. The main purpose here is to ensure that the Sponsor Needs are adequately interpreted and understood. This includes ensuring that all potential errors identified for the Structured Problem Description are avoided. To efficiently achieve this goal, techniques and knowledge from the *application domain* of the model, the area of *requirements engineering*, and *M&S* are advisable.
- During V&V of the Conceptual Model (*Conceptual Model V&V*; column 2.x with x = 1…3 in Figure 14) shall be shown that the Conceptual Model is suitable and correct. The main purpose here is to ensure that the abstraction and idealization of the real system supports and allows the intended examination. This includes ensuring that all potential errors identified for the Conceptual Model are avoided. Knowledge and experience from both the *application domain* of the model and *M&S* are required.
- During V&V of the Formal Model (*Formal Model V&V*; column 3.x with x = 1…4 in Figure 14) shall be shown that the Formal Model is suitable and correct. The main purpose is to ensure that the solution oriented, quantitatively computable description adequately approximates the purpose-oriented, abstract and idealized perception of the real system. This includes ensuring that all potential errors identified for the Formal Model are avoided. Knowledge and techniques from the area of *M&S theory* are required.
- During V&V of the Executable Model (*Executable Model V&V*; column 4.x with x = 1…5 in Figure 14) shall be shown that the Executable Model is suitable and correct. The main purpose here is to ensure that the automatically computable output values approximate (virtual) measurements at the real system under comparable input conditions. This includes ensuring that all potential errors identified for the Executable Model are avoided. *Software engineering* and *programming skills* are required.
- During V&V of the Simulation Results (*Simulation Results V&V*; column 5.x with x = 1…6 in Figure 14) shall be shown that the Simulation Results are suitable and correct. The main purpose here is to ensure that in analogy to experimentation with the real system meaningful Simulation Results are generated. This includes ensuring that all potential errors identified for the Simulation Results are avoided. Knowledge and experience from both the *application domain* of the model and *M&S* are required.

During V&V, systematic examination of each intermediate product by analysis, testing, and test results evaluation under application of mature V&V techniques or techniques assure the desired correctness and suitability. If the intermediate products shall successfully pass the above V&V main phases, their sufficiently high quality is required. To allow the discussion of

V&V activities, it is necessary to clearly separate V&V and model development in the following, and to draw a clear border line: During V&V, failures and errors can be detected – the elimination of identified failures or errors is *not* considered to be part of the V&V process, but to be another (necessary) iteration of model development or experimentation, to achieve the desired quality of the intermediate product (regress in model development).

4.7.4 V&V Sub-Phases and V&V Objectives

The overall V&V goal of each phase to *increase the perceived correctness and suitability* of each intermediate product (and thereby its credibility) is decomposed into objectives, according to the two error classes identified in section 4.7.2. The self-contained examination of each intermediate product is distinguished from the examination of the consistency with previously created intermediate products, yielding two different types of V&V sub-phases. Each V&V main phase is decomposed into

- exactly one sub-phase, in which the absence of *product internal errors* in the intermediate product is demonstrated. (This is symbolized in Figure 14 as a rectangle with a circular arrow, numbered x.1). This type of V&V sub-phase will be referred to as product *internal V&V* in the following.
- one or more additional sub-phases for the pairwise comparison between the intermediate product and previously created intermediate products, demonstrating or confirming the absence of *transformation errors* and propagated errors. (This is symbolized in Figure 14 as a rectangle with a straight arrow from top right to bottom left). This type of V&V sub-phase will be referred to as *transformation V&V* in the following. Here is distinguished between sub-phases that include the earliest possible opportunity for the *detection* of a particular error (dark gray, numbered x.2 in Figure 14), and sub-phases for the *confirmation of absence* of a propagated error, or its delayed detection (light gray, numbered x.3+).

The key concept to manage the complexity of V&V of a model and simulation results is the "divide et impera" concept, here applied according to the three-dimensional model information space introduced in section 4.3.1. For clear separation of objectives, the five V&V main phases are subdivided into 20 sub-phases according to a simple pattern. In the first sub-phase of a V&V main phase, only the intermediate product in its direct context is examined, regardless of any preceding intermediate products. Then with each succeeding V&V sub-phase, those issues characteristic for exactly one preceding intermediate product are added for direct comparison.
In each V&V sub-phase the following issues are addressed:

- **Required stage of model development**: Identification of information about the model that must be available to achieve the goals of the V&V sub-phase.
- **IP counterpart** (transformation V&V only): Intermediate product to which the current intermediate product is compared.
- **Sub-phase goal**: Describes the overall intention of the V&V sub-phase and identifies the V&V sub-phase as a *product internal V&V* sub-phase or as a *transformation* V&V sub-phase. Here also explicitly addressed errors are summarized.
- **Requirements for and consequences of sub-phase skipping**: During several sub-phases the demonstration of the absence of a particular (propagated) error is repeated, using different information and approaches. This implies the existence of redundancy in the complete V&V process. Possible consequences of skipping the sub-phase are pointed out.
- **Impact of error detection on model development**: Here the regress in model development caused by error detection is described, usually including the identification of the phase(s) that must be at least partially repeated.

- **Sub-phase objectives**: Summary of precise objectives that should be achieved during this V&V sub-phase to accomplish the sub-phase goal.
- **Proposed techniques for static analysis**: Summary and short explanation of techniques for static analysis of the documented intermediate product, to reveal potential errors or to make the absence of errors credible. "Static" here means that the model description is not executed (if possible), but analyzed for particular properties that are made explicitly visible and evaluated.
- **Proposed techniques for dynamic analysis**: Summary and short explanation of techniques for test case generation and (if feasible and reasonable) execution of the symbolic model.
- **Proposed techniques for results evaluation**: Summary and short explanation of techniques for the evaluation of input and output data provided for and from the model, respectively. Additionally, V&V techniques sometimes produce results that require further interpretation. Proposals for this subsequent interpretation are given here.
- **External references, leveraged information, and reused V&V output**: Identification of required additional external information and previously created V&V evidence that is (re-) used during this sub-phase for, e.g., comparison purposes.
- **Provided output**: List of V&V results (products) that are gained when executing the proposed V&V techniques.

For each sub-phase, a figure visualizes the dependencies between the objectives, the techniques, their inputs, and the provided outputs (V&V results). Objectives are visualized as boxes with round corners, with light-headed arrows pointing to their associated techniques, depicted as boxes. The required inputs for the application of these techniques, visualized as ellipsoids, point towards the techniques with bold-headed arrows. The same type of arrows points from the techniques to the V&V results (also represented as ellipsoids). An overview of the goals and objectives of the sub-phases is given in Appendix B.

4.7.5 Incremental Credibility Building and Tailoring

In the V&V Triangle, all transformation V&V sub-phases of a row address the question, whether the contents of the associated intermediate product are suitably and correctly reflected by all following intermediate products. The V&V activities recommended in each transformation V&V sub-phase with the objective to "confirm completeness and consistency" (see Appendix A) include the execution of techniques that can be used for indirect confirmation or disprove of the activities conducted during the transformation V&V sub-phase on its left with the objective "demonstrate completeness and consistency". This is achieved by searching for errors that were propagated through the intermediate products examined before. This redundancy can be used to increase the intensity of V&V (both depth and breadth), and thus to increase the credibility established so far, or to tailor the process for more efficient V&V.

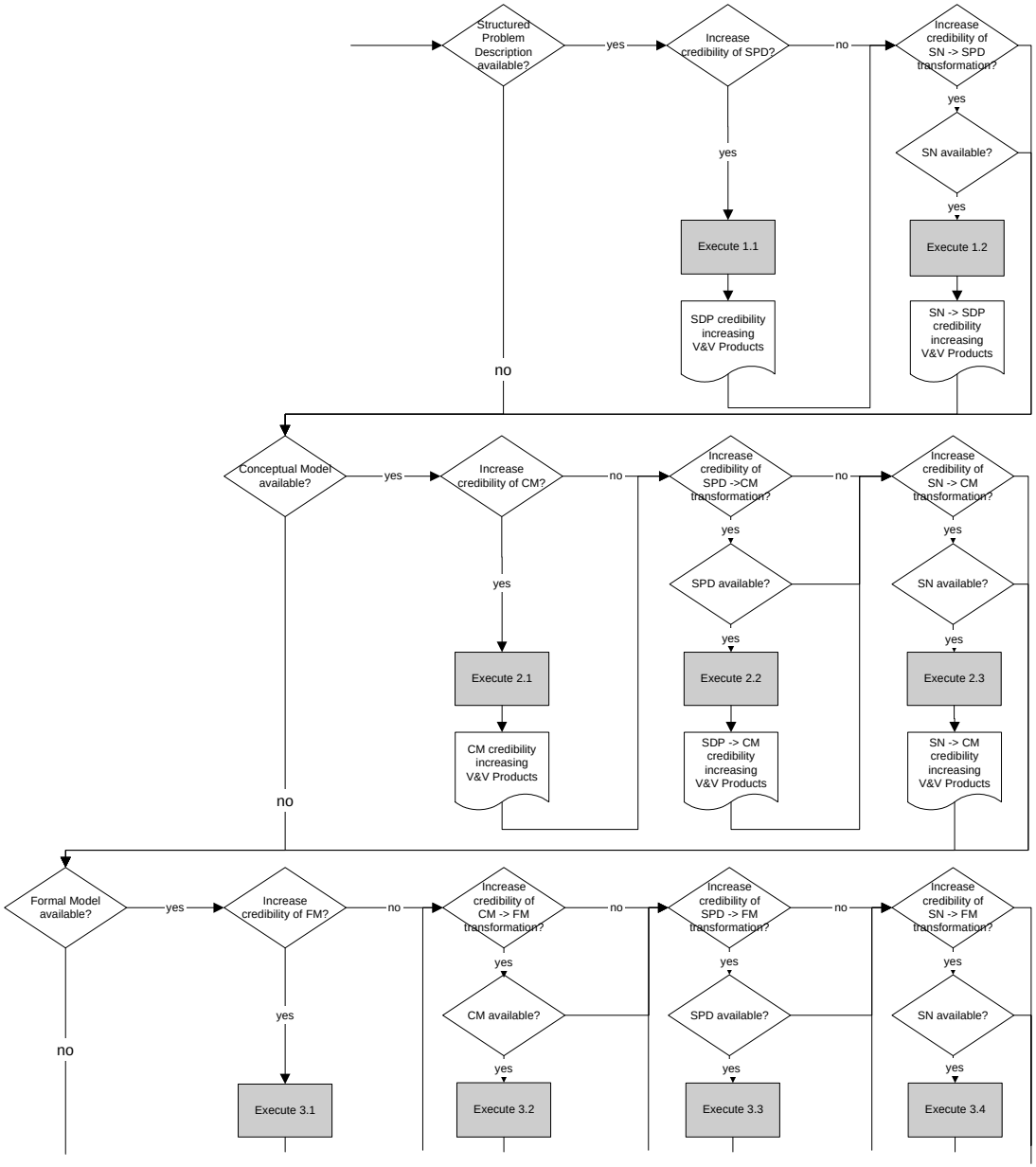
- *Increase the intensity of V&V activities*: By repeating the comparison vs. an intermediate product, this process provides information about the model from a different perspective. It is possible to confirm or disprove previous V&V results.
- *Speed up the V&V process*: Conduct the activities of the most efficient V&V sub-phase and skip all redundant sub-phases. This allows to *tailor* a slim set of V&V activities, at the cost of V&V rigor and intensity.

Example: The Conceptual Model already has been analyzed with respect to its self-consistency and correctness in form (product internal V&V 2.1), and it was demonstrated that the Conceptual Model satisfies the requirements given in the Structured Problem Description (transformation V&V 2.2). If it is found credible that the Formal Model is free of internal

errors (product internal V&V 3.1) and was suitably and correctly transformed from the Conceptual Model (transformation V&V 3.2), it is already implicitly demonstrated that the Formal Model satisfies the contents of the Structured Problem Description. However, executing redundant V&V activities (transformation V&V 3.3) may add additional inspiration to this believe, or may disprove the belief by detection of propagated errors, caused by a previously undetected error in the Conceptual Model.
The assumption of having all intermediate products available is an extreme idealization, almost not given in practice. Usually the V&V budget is also limited. Therefore the selection of the V&V sub-phases that will be run through is influenced by the available system knowledge (see section 4.1) and model knowledge (see section 4.3), time and budget constraints, and the tools available. However, whenever the selection of the V&V sub-phases is a compromise, one must be aware that higher credibility could be achieved by fully exploiting the V&V options presented in this document. Whenever a model is subjected to accreditation, first it can be determined, which of the V&V objectives given in the sub-phases were achieved, and which evidence was created (and which not). The decision, whether the power of the V&V results is sufficient to inspire the necessary belief in the correctness and suitability, remains to the sponsor, or a sponsor representative.
Credibility assessments, which are based on a (standardized) procedure like the V&V Triangle, are comparable (to a certain degree). Evaluating the results of the documentation of the sub-aims, a more objective judgment with respect to the overall validity of the model can be made.
The V&V Triangle supports V&V planning and V&V implementation:

- When using the V&V Triangle for V&V planning, one first determines, which intermediate products are available or will become available during model development (model knowledge). Then, referring to the objectives of each V&V sub-phase, the risk incident to the use of the model or simulation results, the errors addressed during each sub-phase, the required reference information (system knowledge), and the expectable V&V products, one determines, which sub-phases have to be implemented for the purpose of creating belief inspiring material with sufficient power. By choosing a set of V&V sub-phases (and thereby pre-determining the coverage of V&V objectives given in the V&V Triangle), the rigor of the V&V activities is outlined.
- For implementation, as soon as the first scheduled intermediate products are provided, one can start with the first scheduled V&V sub-phase to identify the first objective. The intensity of the V&V activities depends on the preciseness of the evaluated model information and the thoroughness of the V&V techniques implementation. Whenever the activities that are recommended in the first scheduled sub-phase are completed and documented, it may become necessary to repeat one or more steps of model development due to detected incorrectness or unsuitability of the examined intermediate product. (This yields a new version of the first intermediate product and initiates the repetition of the first scheduled sub-phase.) Otherwise, the V&V agent starts with the activities recommended for the next scheduled V&V sub-phase, to achieve the next V&V objective. When there are no more scheduled sub-phases in a column left, the V&V of the current version of the intermediate product is concluded, and the next intermediate product is subjected to examination as soon as available. If planned V&V products cannot be generated during V&V implementation, it becomes necessary to adapt the V&V plan.

SN: Sponsor Needs
SPD: Structured Problem Description
CM: Conceptual Model
FM: Formal Model
EM: Executable Model
SR: Simulation Results



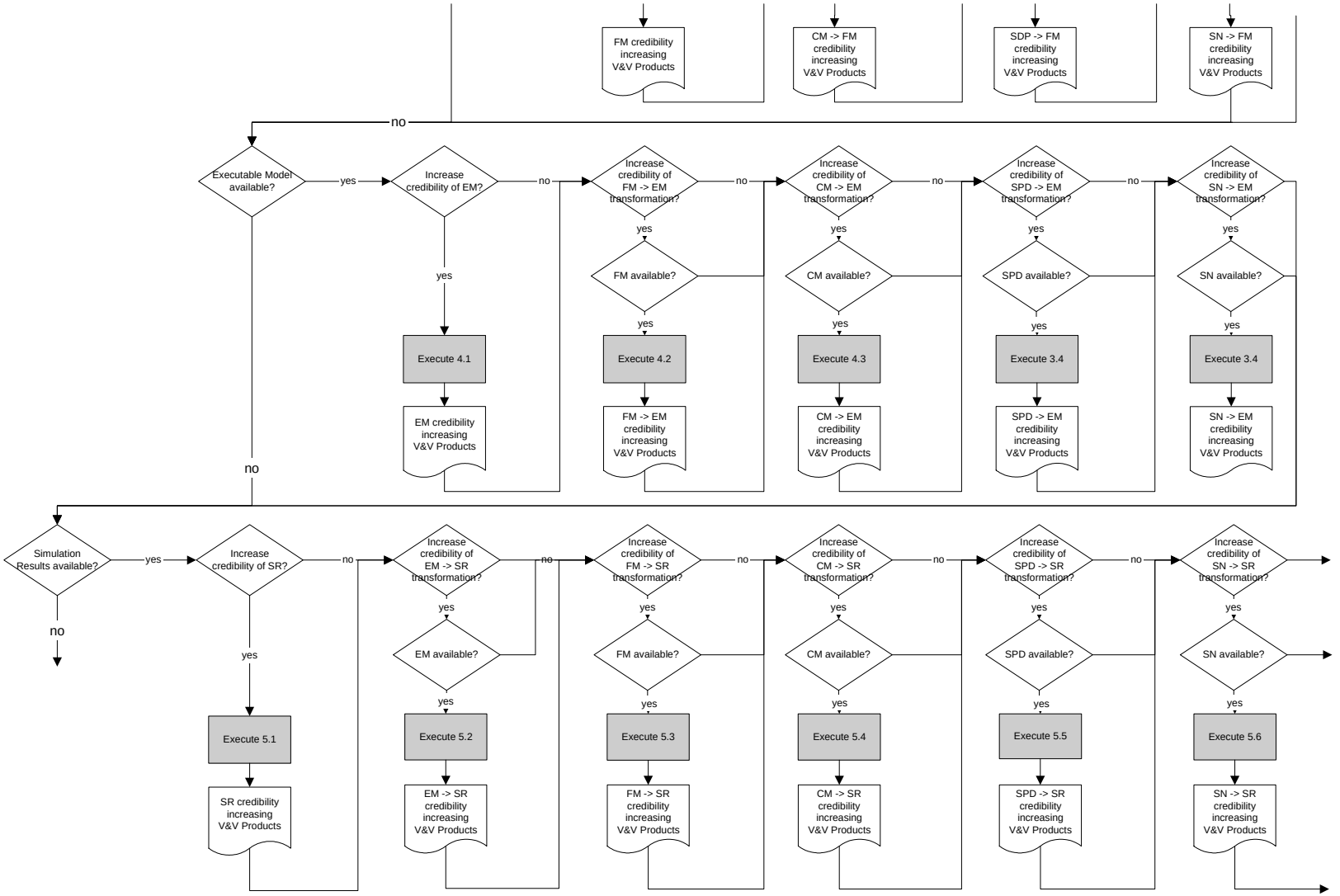


Figure 15: Procedure for planning based on the V&V Triangle.

Figure 16 visualizes a V&V process tailored from the V&V Triangle, yielding a model development and execution process in V-form (see section 1.4.3). All intermediate products are examined for correctness in form and self-consistency (sub-phases x.1 with x = 1…5), and completeness and consistency with respect to their predecessor (sub-phases x.2 with x = 1…5). But only two V&V sub-phases with the aim to confirm previously created V&V evidence are executed, which minimizes the additional effort. However, the detection of any violation of the Structured Problem Description undetected in sub-phase 2.2 is delayed to sub-phase 4.4, and any failure in specifying the Sponsor Needs undetected in sub-phase 1.2 propagates through all intermediate products until discovered during the evaluation of the Simulation Results.

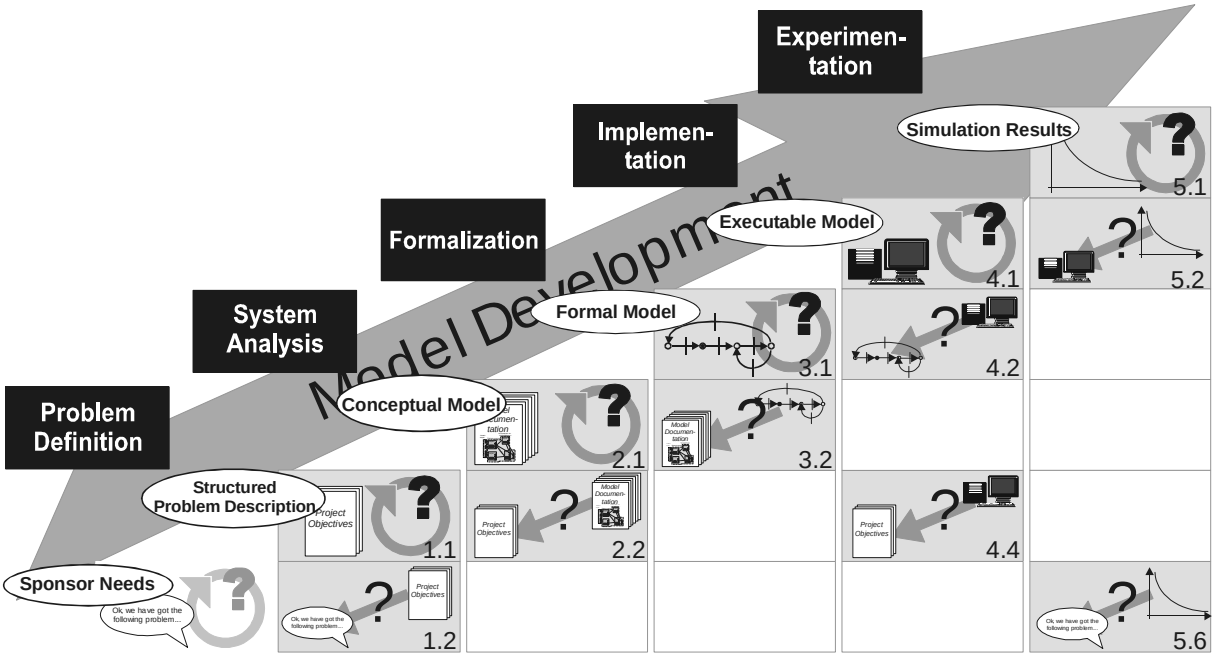


Figure 16: A tailored V&V process in V-form

This concludes the introduction of the ideas and concepts of the V&V Triangle. The detailed specification of the anticipated developmental activities and their failures, the intermediate products, the expected errors, and the V&V sub-phases, including V&V objectives, V&V techniques, and V&V evidence, is provided in Appendix A. An overview of the V&V objectives and the generated V&V evidence is given in Appendix B.

5 EVALUATION OF THE V&V TRIANGLE

There are 74 different objectives identified within the V&V Triangle, which may be achieved during V&V (which is then "exhaustive" V&V with respect to the V&V Triangle). However, the V&V Triangle does not propose exhaustive, static V&V in form of a "cookbook approach". Its intention is not to force those responsible for conducting V&V into the cumbersome task of achieving all V&V objectives and to substantiate all of them with evidence. The opposite is the case – the V&V Triangle supports the selection of *harmonized* V&V objectives, and points out their dependencies, to enhance the use of as many of their synergetic effects as possible.
The V&V Triangle covers most of the known operational approaches to V&V, including *requirements tracing* and *face validation*, which was shown to be true for at least the M&S sample project. In this section, *paths* taken through the V&V Triangle are described, which illustrate the application of the approach introduced in section 4.7. Also *enablers* are presented that allow the selection of the paths. The dependencies between the V&V techniques within a path are explicitly pointed out. The relationships between the different paths taken become obvious, as the V&V sub-phases processed on each path are graphically highlighted.
The paths that are described and evaluated in the following include

- the *path of documentation correctness in form and internal consistency*, which focuses on the documentation of the intermediate products,
- the *path of behavior propositions violation detection*, which focuses on the identification of violations of formally specified behavior propositions,
- the *path of human review*, which focuses on techniques that allow human experts to make deliberate statements about the correctness and suitability of a model or simulation results.

5.1 The Sample M&S Project

For the purposes of illustration and evaluation of the V&V Triangle, a small, but not trivial performance model of a road intersection was created. A brief summary of the model and its use is anticipated here, the detailed documentation of the intermediate products created during model development is documented in Appendix C.
The model was created to facilitate the comparison of different lights control strategies for the Munich road intersection "An der Point Süd", without impacting the traffic flow in the road intersection, taking into account the lights strategy and the density of traffic from the three different directions. This performance analysis task is purely fictive and has no real origin – nevertheless, the chosen road intersection exists close to the "Neue Messe München" in the east of Munich. Most information of the real system originates from official sources (the "Traffic Regulation Office"), as documented in Appendix C. As result of the system analysis phase, three major sub-systems were identified, as they are:

- The *intersection geometry*, which models the intersection layout, the lanes, and the positions of the lights;

- the *lights control unit*, which models the switching logic of the traffic lights;
- the *vehicles*, which model the participants in road traffic and their behavior in the road intersection.

The Conceptual Model is formalized using DEVS [Zeigler, Praehofer, and Kim 2000], preparing it for discrete event simulation, and finally implemented in Wolverine's simulation environment SLX [Henriksen 1998]. 81 experiments were executed using the intersection model, yielding average and accumulated alive (instantiation until destruction) and waiting times (velocity equal or less than 1m/s) of the vehicles arriving from the different directions.
In-house development of the model facilitated the documentation of all (immediately available) intermediate products according to the required contents summaries provided in Appendix A. Whenever additional information about an intermediate product was required, the contents summaries were modified or extended as appropriate.

5.2 Documentation Correctness in Form and Internal Consistency

Understanding, duplicating, and assessing the knowledge and information available as a foundation for a model are among the most important aspects of V&V of models and simulation results. Model documentation provides insight into the model, increases its transparency, and thereby becomes an indispensable prerequisite for V&V. Regardless of the actual contents that are documented and their meaning for the application domain, requirements for "good" model documentation exist that are valid for all cases of model documentation. The application of techniques for checking that these documentation requirements are met is found in this section.
Within the chosen model development process (section 1.4.6), formal-mathematical aspects (documented in the Formal Model) and technical aspects (documented in the Executable Model) are added incrementally to the abstracted and idealized representation of the real system (documented in the Conceptual Model). But the separated documentation of the conceptual, formal, and technical aspects of a model brings another challenge; documentation that is not complete and consistent may become dangerous, if it implies assumptions about the (executable) model that are not true. To be complete and consistent, each intermediate product must satisfy all requirements defined by its predecessor, and provide all information required for further model development.
To facilitate V&V as described by the V&V Triangle for the M&S sample project (as documented in Appendix C), the complete model documentation includes the documentation of the Sponsor Needs, the Structured Problem Description, the Conceptual Model, the Formal Model, the Executable Model, and the Simulation Results. Whenever a new intermediate product was subjected to V&V, documentation correctness in form checking was conducted first. The key features of the V&V Triangle to support the availability of intermediate products, which are comprehensible, correct in form, complete, and consistent are

- documentation requirements, forms, and templates,
- techniques for completeness and consistency checking, and
- modeling formalisms and formal specification techniques.

These key features strongly encourage tool support.

5.2.1 Background – Structured Documents

Not only to support V&V, but also to enhance maintenance and re-use, all documentation efforts should result in comprehensible, structured documents. *Structured documents*, which are automatically generated from a data base, are successfully applied to the documentation of

technical products, which often are subjected to modification. The main advantages of structured documents include

- their hierarchical organization, which allows tool supported efficient browsing through the documents, and
- their significantly simplified maintenance by automatic generation on demand from the data base, which stores their contents.

As on a sufficiently high level of abstraction the required structure and type of contents of the documentation of all models and simulation results are identical, the concept of structured documents was transferred to model documentation, with the aim to populate the three-dimensional model information space, according to the content requirements given in the sections 2.2, 3.2, 4.2, 5.2, and 6.2 of Appendix A. The V&V techniques evaluated in the following focus on the completeness, internal consistency, and pairwise consistency of the documentation of each intermediate product.

5.2.2 The Enablers: Check Lists, Templates, Dependencies Matrices, and Formalisms

For model development it is crucial that all potential future developmental issues are at least raised (even, if left deliberately unresolved) as early as possible during the model development process, otherwise undesired degrees of freedom are introduced into model development.

Ideally, the documentation of each intermediate product includes all required contents in the desired format (is *correct in form*), which are not contradictory among themselves (*internally consistent*), and meets all the requirements defined by the contents of the preceding intermediate product (i.e. the contents of the intermediate product are consistent with the contents of the predecessor, and reflect them completely). Correctness in form is always determined with respect to a given form (or formalism), which was designed to cover all required pieces of information. To require *correctness in form* enhances completeness in content and consistency, the availability of an adequate form assumed.

To ensure that nothing important is forgotten, *checklists* or *templates* are helpful. From the checklist or template a matrix can be created, which identifies those contents that need to be consistent within the checklist or template, and with the contents of other checklists and templates (*dependency matrix*). When using checklists, templates, or consistency matrices for checking correctness in form, completeness, and consistency, the result of their application are *completed* check lists, *completed* templates, or *completed* dependency (*traceability* or *consistency*) matrices.

5.2.2.1 Checklists

For each intermediate product there is a list of required contents to provide the desired insight into the model as those documented in the Appendix A, sections 2.2, 3.2, 4.2, 5.2, and 6.2. Based on these lists of required contents, checklists are created with a checkbox available for each required content (Example: FEDEP Checklists, [Defense Modeling and Simulation Office 1999b]). For completion of a checklist, someone reviews the available intermediate product and checks the boxes, if justified (reviewer's opinion!). The main effort is to review the available documents, whereas the effort for checking the boxes can be considered as negligible. No in-depth examination is conducted when using checklists, it is only assured that there is any information as required by the check box available at all. (For example, if the documentation requirements contain the need for an unique identifier for configuration control, it is only checked, whether there is such an identifier in the correct format. The question of its uniqueness remains unexamined.) The benefits of applying this technique depend on the maturity of the list of required contents (i.e., the underlying expert knowledge) and the reliability and skill of the reviewer, as the check list only serves as evidence for the fact that someone

actually reviewed the intermediate product for the contents marked as “checked”. The power of inspiring believe may be increased by referencing associated sections or paragraphs in the documentation for each checkbox, but this increases the checking effort significantly.
In the context of this thesis, checklists where applied only implicitly. With the opportunity to document the M&S sample project in any desired format, the stronger approach of using *templates* was chosen.

5.2.2.2 Forms or Templates

For each intermediate product, a predefined form or template is provided. During the M&S sample project, the intermediate products were created based on their associated form. This is a constructive technique for error avoidance; now it becomes obvious already during model development, whether there is the required information available in each cell of the form or template, or not. Thus, checking for correctness in form is reduced to assessing the contents in each cell, without the need to find the desired information somewhere in a document (as with the checklist approach). The main effort during V&V consists of reviewing the completed template and to check it for empty cells, cells with entries incorrect in format, and fields with entries that are obviously wrong. If the reviewer is familiar with the template, the reviewing effort may even be lower than reviewing a document structured in an unknown format. By looking through the completed templates, every reviewer can quickly reassure himself that there are the required entries in the fields, or not (in contrast to the checklist approach, where confirmation of the first “checker’s” work means repetition of the complete checking procedure).

5.2.2.3 Consistency Matrices

For intermediate product internal or pairwise consistency checking, a consistency rules set is created from the template(s) of the intermediate product(s) involved. The creation of the rules set includes the identification of contents that must be consistent with each other (in the M&S sample project given as *dependency matrices*), and one or more rules that must not be violated by these contents. However, the less formal the specification of the contents examined for inconsistency, the more difficult becomes the specification of rules, which in general leads to a high involvement of human experts for assessment of consistency especially in the early phases of model development.
Consistency checking applied *internally* for an intermediate product helps to detect inconsistencies between its various contents (*internal consistency*), and results in a completed *intermediate product internal consistency matrix*. Consistency checking applied to pairs of intermediate products creates a trace between the contents of the intermediate product and the contents of its predecessors (*pairwise consistency*), which results in a completed *IP-IP traceability matrix*. Among the advantages of the use of traceability matrices is that incompleteness with respect to requirements or model contents can be detected by tracing each entry of an intermediate product to the entries of its predecessors (*requirements tracing*). Good experience was made with requirements tracing in the software domain, and there was no need to copy them for M&S V&V. Other techniques were explored for “inter-IP-consistency checking”, as documented in sections 5.3 and 5.4.

5.2.2.4 Modeling Formalisms

As pointed out in section 4.1, during the chosen model development process a model is formalized in a solution-oriented, unambiguous form to prepare its solution, using a suitable modeling paradigm or formalism. Modeling formalisms provide powerful support to assess consistency and completeness of the (formal) model specification. They own a syntax and semantics specification with which the formalized model must be compliant. The syntax and semantic rules implicitly contain requirements concerning correctness in form, consistency,

and completeness, which means that inconsistency and incompleteness can be revealed indirectly and efficiently by the detection of syntax or semantic errors. With an Formal Model specification available, intermediate product internal consistency and completeness checking is replaced by syntax and semantic analysis with a higher potential for automation and thus a lower likelihood of failure.

5.2.2.5 Tools Support

The use of forms or templates and the desire for correctness in form encourage recording of the contents of the intermediate products in a database, which is organized and typed according to the form or template. Then, for documentation of the intermediate products, electronic forms are filled in at the computer screen, and the information is stored in an underlying data base. Thereby all writing (and reading) access to the data base is directly controlled, and rigorous checks for correctness in form, completeness, and consistency are automatically performed or initiated whenever an entry is made or modified. As soon as an entry is made, it is checked that it satisfies the formal requirements and characteristics of its field. Consistency checks can be guided by dialogues ("wizards") based on dependency matrices and the consistency rules set.

As the dependencies between the various intermediate product contents and the associated consistency rules are most likely to become quite complicated, tool support is strongly recommended for guidance through consistency checking. In [Morales and Moulding 2000] the efficiency of an automated consistency checker for SOM's and the FOM of HLA federations is demonstrated. In the context of this thesis, two prototypes of tools that support the structured documentation of models and simulation results, checks for correctness in form, completeness, and consistency were designed and implemented. For details please refer to [Schultheiß 2001] and [Moritz 2003].

Modeling formalisms also enhance tool support, and only develop their full potential integrated into a development environment, where checks for syntactical and semantic correctness, and thereby correctness in form, consistency, and completeness can be automated. Formal specification techniques used in the context of the M&S sample project are PLTL (section 5.3.2), DEVS [Zeigler, Praehofer, and Kim 2000] as modeling formalism, and the simulation environment SLX [Henriksen 1998] for the implementation of the Executable Model. The PLTL formulas were parsed successfully by VIOLADE [Brade and Waldner 2003], as already discussed in section 5.3.8. The Executable Model specification in SLX was created in the SLX development environment with computer aid for (automated) syntax and semantic analysis available.

5.2.3 Planning with the V&V Triangle

In the V&V Triangle, the examination of correctness in form and internal consistency is always conducted in the first V&V sub-phase (x.1) associated with each intermediate product. Consistency with the preceding intermediate product and completeness is always demonstrated in the second sub-phase (x.2). (This demonstration is omitted in the M&S sample project, where the focus lies on the exploitation of less generic techniques for demonstration of pairwise consistency.) The V&V objective "confirmation of completeness and consistency" with earlier intermediate product (sub-phases x.3+) needs only to be achieved, if additional evidence for the traceability of the particular contents is required, or if the directly preceding intermediate product is not available. (For example, if there is no Formal Model, consistency between Conceptual Model and Executable Model cannot be shown by tracing contents of the Executable Model through the Formal Model to the Conceptual Model.)

With all intermediate products available, and the assumption that in the M&S sample project no traceability matrix based confirmation for completeness and consistency is required on the path of correctness in form, the V&V sub-phases x.1 are visited, as sketched in Figure 17. All

V&V objectives and V&V techniques are taken from the V&V sub-phase specification documented in Appendix A. With the opportunity to determine the format of intermediate product documentation, the Formal Model specified in DEVS, and the Executable Model implemented in SLX, the following V&V plan was created:

- Require intermediate product documentation according to the associated lists of required contents defined in the Appendix A, sections 2.2, 3.2, 4.2, 5.2, and 6.2.
- Sub-phase 1.1 (Structured Problem Description)
  - Demonstrate correctness in form: Review intermediate product documentation, confirm compliance with the associated template;
  - Demonstrate self-consistency: Create internal dependency matrix from intermediate product template, check dependent contents for consistency using expert opinion, complete intermediate product internal consistency matrix.
- Sub-phase 2.1 (Conceptual Model):
  - Demonstrate correctness in form and self-consistency: Perform correctness in form checking, I/O parameters declaration analysis, interface analysis, and dimension unit analysis.
- Sub-phase 3.1 (Formal Model):
  - Demonstrate correctness in form and self-consistency: Perform correctness in form checking and syntax and semantic analysis of the DEVS specification.
- Sub-phase 4.1 (Executable Model):
  - Demonstrate correctness in form and self-consistency: Perform correctness in form checking and syntax and semantic analysis of the SLX code.
- Sub-phase 5.1 (Simulation Results):
  - Demonstrate correctness in form: Review intermediate product documentation, confirm compliance with the associated template.

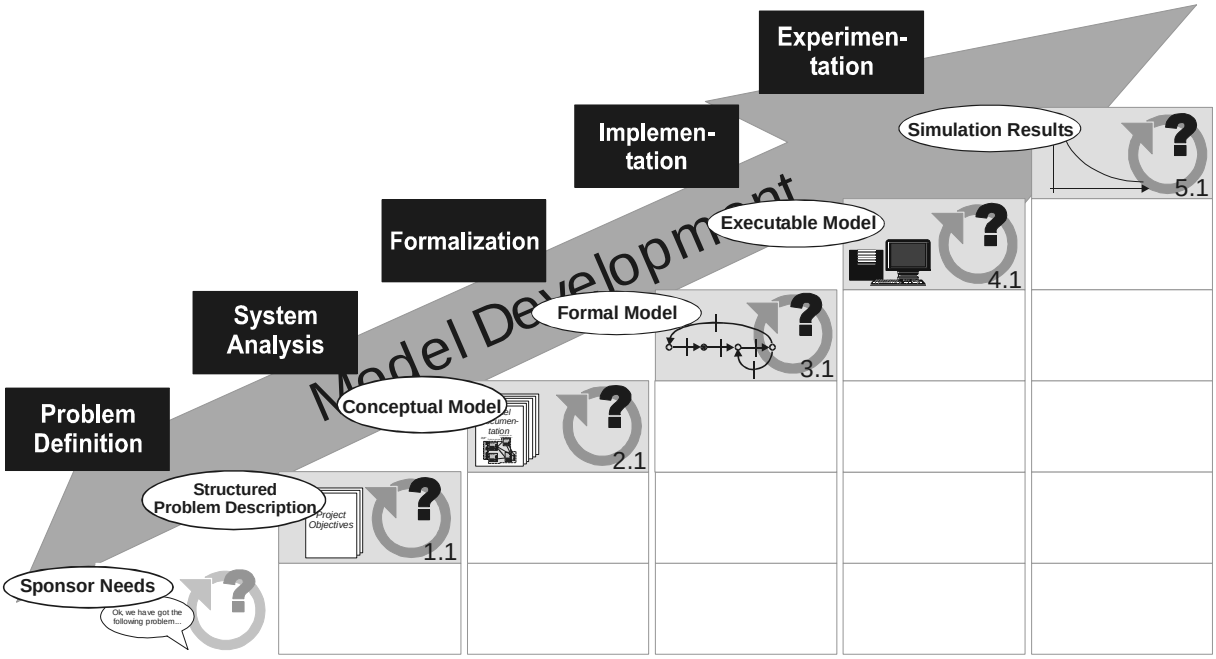


Figure 17: Path of correctness in form and internal consistency

5.2.4 Sub-Phase 1.1: Correctness in Form and Self-Consistency

Here the focus of the V&V activities lies on the demonstration that the documentation of the Structured Problem Description is compliant with the Structured Problem Description template (which requires the recording of all information relevant for further model develop-

ment). To make the demonstration of compliance as easy as possible, the Structured Problem Description of the M&S sample project was modified in an iterative process until it was documented according to the Structured Problem Description template defined in Appendix A, section 2.2. Although the Structured Problem Description of the M&S sample project is not completely compliant with the template structure, all relevant contents for further model development are provided. Comparison of the dependent contents did not yield inconsistencies.
It is essential for the success of an M&S project that the Structured Problem Description covers *all* Sponsor Needs correctly in content, but this belongs to sub-phase 1.2 and is beyond the path of correctness in form and internal consistency. Due to the fictive nature of the M&S sample project, the sponsor's real needs were assumed to be completely documented in the Sponsor Needs documentation, and no real review took place (which is considered to be careless in a real M&S project).

5.2.5 Sub-Phase 2.1: Correctness in Form and Self-Consistency

For the Conceptual Model documentation, correctness in form is also demonstrated by a check for compliance with the Conceptual Model template. To demonstrate internal consistency of the Conceptual Model, due to the exploratory nature of the M&S sample project the creation of a Conceptual Model internal dependency matrix was omitted and replaced (instead of supplemented) by other, more specialized V&V activities as *I/O parameter declaration analysis*, *interaction analysis*, and *dimension unit analysis*.

5.2.5.1 Correctness in Form Checking

The documentation requirements given in section 3.2 are used as a reference for the determination of correctness in form of the Conceptual Model. To make this as easy as possible, the Conceptual Model was documented in an iterative process according to the Conceptual Model template (Appendix A).

5.2.5.2 I/O Parameter Declaration Analysis

This analysis technique is based on the precondition that the targeted model is hierarchically organized in submodels. The interface of each conceptual submodel must be consistent with its internal functionality. Due to the informal nature of the Conceptual Model, this analysis activity cannot be included into automated syntax or semantic analysis and needs to be performed "manually". The procedure for I/O parameter declaration checking is straight forward:

1. Identify all functions (including state transition conditions or other functional I/O dependencies) in each conceptual submodel;
2. Identify all parameters involved in the functions, categorize them as input, state, and output parameters;
3. Check, whether all input and output parameters are declared in the associated submodel interface as input parameters or output parameters, respectively. As a side effect, confirmation of the internal consistency of the state variables declaration is supported.

This technique helped to reveal the required input and output of each submodel in the M&S sample project, and thereby had a constructive impact on the iterative identification of the conceptual submodel borders and overall Conceptual Model formulation. However, the informal nature of the Conceptual Model prohibits the automation of the I/O parameters declaration analysis, which implies that a rigorous, computer aided check must be delayed until the Formal Model stage.

5.2.5.3 Interaction Analysis

In a hierarchically organized Conceptual Model, the "interfaces" between the conceptual submodels need to be consistent, and require special consideration during the demonstration of internal consistency. (This is of great importance for reused and "composed" conceptual submodels, but also the exchange of a submodel in the M&S sample project revealed the importance of syntactically and semantically consistent submodel interfaces.) After confirmation that the conceptual submodel interfaces actually reflect their I/O needs, interaction analysis can reveal inconsistencies in submodel interaction. "Interaction analysis" for the Conceptual Model was derived from interface analysis as used in Software Engineering [Beizer 1990], and slightly modified. The implementation of the technique is straight forward:

1. Identify all documented submodels within the submodel hierarchy;
2. Identify all documented interactions between submodels;
3. Analyze, whether the defined interfaces and submodel responses allow the desired interactions.

Actually this is an extremely specialized form of manual or mental execution, where one concentrates on the interaction of two or more submodels. By executing interaction analysis, in the M&S sample project several potential concurrency problems were detected before they could propagate to the Formal Model or Executable Model, including the need for vehicle-vehicle interaction of memorization of the previous vehicle attributes after position, velocity, and acceleration update.

5.2.5.4 Dimension Unit Analysis

This is actually a highly specialized form of the semantic analysis, which requires mathematical equations with dimension units (e.g., *m* for meters as physical dimension unit, or € as currency) assigned to the parameters. By applying mathematical operations to the dimension units it is determined, whether the correct resulting dimension unit is calculated. This technique was used to demonstrate the correctness of the kinetic equations used for vehicle movement in the M&S sample project.

5.2.6 Sub-Phase 3.1: Correctness in Form and Self-Consistency

In the M&S sample project, the Formal Model documentation consists of the model specification in Parallel DEVS [Zeigler, Praehofer, and Kim 2000]. Checking correctness in form and self-consistency of the Formal Model with respect to the Formal Model documentation template itself is less important in this phase, as the syntax and semantics of a well-defined modeling formalism contain all requirements for complete and consistent model specification. As soon as a formalism for model specification with well-defined syntax and semantics is available, the violation of form or internal consistency requirements usually results in a syntax or semantic error. However, it needs to be ensured that the supplemental information, which is also a part of the Formal Model documentation is consistent with the Formal Model itself, which can be performed by using the associated template and dependency matrix. (For example, the referenced modeling formalism should be identical with the formalism used for formal specification of the model contents.)

5.2.6.1 Demonstrate Correctness in Form

The documentation requirements given in section 4.2 are used as reference for determination of formal correctness of the Formal Model documentation. To make this as easy as possible, the Formal Model was documented in an iterative process according to the Formal Model template (Appendix A). The cumbersome completion of the Formal Model internal dependency matrix was omitted due to the exploratory nature of the M&S sample project.

5.2.6.2 Syntax Analysis

With Parallel DEVS as the formalism used for Formal Model specification, a stable foundation for rigorous syntax analysis is available. There is an unambiguous formal specification for DEVS models [Zeigler, Praehofer, and Kim 2000]: All atomic submodels are defined as (mathematical) 7-tuples, with a set of input events, a set of output events, a set of states, an internal transition function, an external transition function, a concurrent events resolution function, an output function, and a time advance function. To couple atomic models to coupled models, new sets are created, and again this needs to be formalized using mathematical declarations and equations. For the M&S sample project it was shown that only (lexical) tokens allowed by DEVS were used, and that they were combined according to the underlying grammar.

This was performed manually, as the effort of transforming the Formal Model specified in DEVS into a machine-readable form (e.g., ASCII in XML format) is high. However, this effort could be significantly reduced, if the Formal Model was developed using a DEVS specification tool, which was not available at the time it was required. (DEVSJAVA [DEVSJAVA 2.7 2003] and CD++ [Glinsky and Wainer 2002] were considered to be too closely tied to an implementation solution.) A diploma thesis was initiated to close this gap [Schäfer 2003] by the design and implementation of a DEVS-based visual modeling environment (see also section 6.10).

5.2.6.3 Semantic Analysis

For the DEVS model of the M&S sample project also semantic analysis was conducted. For DEVS this is the confirmation

- that the value domain of an output port of an atomic or coupled DEVS is equal to or contained in the value domain of the associated input port of a directly interconnected DEVS (internal coupling), or
- that the value domain of an input or output port of an atomic or coupled DEVS is equal to the value domain of the associated input or output port of a contained atomic or coupled DEVS, respectively (external input and output coupling),

which was shown to be true for all interconnected and embedded submodels in the M&S sample project.

5.2.7 Sub-Phase 4.1: Correctness in Form and Self-Consistency

In the M&S sample project, the Executable Model is available as SLX code in the SLX simulation environment. As for the Formal Model documentation, the Executable Model documentation template is only required to assess the availability of the desired supplemental information. The demonstration of the correctness in form of the Executable Model itself is more efficiently performed by automated syntax and semantic analysis.

5.2.7.1 Demonstrate Correctness in Form

To make the demonstration of correctness in form as easy as possible, the Executable Model documentation was developed according to the Executable Model documentation template specified in section 5.2 of Appendix A. The demonstration of self-consistency was omitted due to the exploratory nature of the M&S sample project.

5.2.7.2 Syntax and Semantic Analysis

As the Executable Model in this case is a model that is used for computer-based simulation, it needs to satisfy the same requirements as any piece of software, concerning the unambiguousness of its specification. Execution on a computer includes compilation or interpretation,

which implies the existence of a syntax and semantic analyzer that was used to confirm the syntactical and semantic correctness of the Executable Model.

5.2.8 Sub-Phase 5.1: Correctness in Form and Self-Consistency

In the M&S sample project, for the Simulation Results documentation the demonstration of correctness in form using the Simulation Results documentation template is again of major importance, as no formalism is used for the documentation of the experiment setup and the Simulation Results. The explicit checking of correctness in form is extremely important, as the description of the experiment setup and the data acquisition process usually is informal.

5.2.8.1 Demonstrate Correctness in Form

To make the demonstration of correctness in form as easy as possible, the Executable Model documentation was developed according to the Executable Model documentation template specified in section 6.2 of Appendix A.

5.2.8.2 Experimentation Documentation Internal Consistency Checking

This is a specialized form of the demonstration of correctness in form, which concentrates on the documentation of the experiments. For each documented experiment, not only the Simulation Results need to be documented, but also a detailed description of the complete experiment setup is required. For the M&S sample project this was shown to be true – although the Simulation Results are due to their extend only documented in aggregated form, the raw model output is electronically available.

5.2.9 Concluding Remarks

Diverse representations of a model are required to understand, master, and execute an Executable Model on a computer platform. Completeness and consistency of these representations are essential to allow the transfer of knowledge gained by the use of one representation to another. During the M&S sample project and in [Brecht and Riepl 2002] it was shown that the lists of required contents provided in Appendix A encourage the modeler to create documentation that is useful for subsequent V&V activities. The need to document the required contents also encouraged the model developers to write down their thoughts, a procedure that, simply by executing it, revealed gaps and inconsistencies in the (apparently stable) mental constructs.

Nevertheless, models quickly become quite complex systems themselves, and completeness and consistency can only be ensured with an acceptable effort by the use of supporting tools. Without suitable tools the creation and documentation of consistency matrices and traceability matrices is cumbersome, inefficient, and error prone.

The main problems with the model documentation included:

- Changes in either representation of the model (Conceptual Model, Formal Model, or Executable Model) needed to be manually transferred to the other model representations. This lead to inconsistencies between the different model representations, which becomes obvious after intensive study of Appendix C.
- The manual check for internal consistency allowed the acceptance of unacceptable inconsistencies during the review process. Computer support is recommended to enforce a more rigorous consistency check.

As previously indicated, the intermediate product documentation templates can be used for the organization of a data base, which allows the electronic storage of the model contents. The use of commonly accepted standards as XML or ODBC assumed, the access to this data base and the check for consistency can be controlled by specialized, exchangeable tools. In addition, this data base can be used to populate a model repository, which holds in-depth informa-

tion about a number of models in a standardized format (see also section 6.10). The result of following the path of documentation correctness in form should be model documentation that makes the model sufficiently transparent for following content-oriented V&V activities.

5.3 Automatic Detection of Behavior Proposition Violation

By definition, simulation includes the execution of a model over time (see section 1.3.2). When an Structured Problem Description is created, often implicit model behavior requirements are anticipated. Then, as soon as the Conceptual Model is completed, in the modeler's mind there is a clear imagination of the *desired behavior* of at least its submodels. However, desired behavior may not equal the *observed behavior* of the executable submodels, as long as undetected errors remain. This becomes delicate, if the undesired Executable Model behavior remains unobserved until it is too late and damage is caused.
In the following, the thesis concentrates on discrete state transition models (see also section 1.2.4). *Model behavior* is the change of the internal state of the model and its output, depending on its previous state and input. Typical undesired behavior that occurs in discrete state transition models includes:

- *Undefined submodel state*: The value of an attribute, or the value combination of two or more attributes reflect undefined submodel behavior. (Example: An object representing a vehicle in a road intersection, with an acceleration of 35 m/s^2).
- *Undefined state transition*: An object changes states in an undefined way. (Example: If for an object "vehicle" there is a state "accelerating", behavior specification may require that the vehicle in state "standing" must pass through the state "accelerating" to reach the state "cruising". Then a direct transition from "standing" to "cruising" is undefined.)
- *Missing effect*: A state transition that is caused by the occurrence of a particular event is missing. Example: (The state transition "change of color" of traffic lights from red to green should cause the state transition from "standing" to "accelerating" for the waiting vehicle.)
- *Missing cause*: A state transition without any specified cause (state or state transition prior to it). Example: The vehicle switches from "cruising" to "decelerating" although the lights stay green and there is no slower vehicle in front of it.

Those types of undesired behavior may be detected by subject matter experts through visualization as proposed for the sub-phases 4.1, 4.3, 5.1, and 5.4, but to do so for a significant number of test runs or test experiments requires patience and a lot of time. Especially, if the undesired behavior occurs rarely, it may be hardly detectable by human review. Statistical techniques sometimes allow to backtrack a violation from statistical test results, if the behavior violation occurs frequently enough to impact statistical analysis of model output. However, the *formal specification of desired behavior* derived from the Structured Problem Description (V&V sub-phases 1.1, 1.2) and the Conceptual Model (V&V sub-phases 2.1, 2.2) as explicitly expressed behavior propositions allows direct detection of behavior violations.

5.3.1 Background – Formal Specification of Software Requirements

The idea of formal specification of desired behavior extracted from a "natural language specification" is successfully applied as *Cause-Effect Graphing* for software. A comprehensive introduction to Cause-Effect Graphing is given in [Myers 1979]. Based on the assumption that exhaustive testing of software is rarely possible, Cause-Effect Graphing is a technique that aids in selecting in a systematic way a high-yield set of test cases, and supports the identification of incompleteness or ambiguities within the software specification.

A Cause-Effect Graph is a formal description into which a natural-language specification is translated. It is based on Boolean logic, which is extended by constraint operators. To simplify the creation of the Boolean expressions, graphical representations for the Boolean operators "identity", "not", "or", and "and" are provided, which allows the creation of a cause-effect graph according to the following process:

1. Identify all causes and effects in the natural language specification: A cause is a distinct input condition or an equivalence class of input conditions. An effect is an output condition or a system state change.
2. Analyze the semantic content of the specification and transform it into a Boolean graph, which links the causes and the effects.

According to [Myers 1979], already the transformation from the natural language specification of the software into the formal specification reveals numerous ambiguities and errors. The Cause-Effect-Graph can be efficiently reused, if one needs to test, whether the "and" or the "or" interconnections of effects yield the desired causes in the implemented software. The Boolean "and" and "or" allow to skip several input combinations, respectively. Myers states that it is sufficient for the testing of an effect, which is to be triggered by the parallel occurrence of several causes, just to check, whether the effect occurs when all causes are given, and to ensure that the absence of any single cause does not result in the effect in this case. All cause combinations, where more than one cause (but less then all) are not true, do not need to be explicitly tested, they are implicitly covered by the previously introduced test cases. Assume there are n causes, which may be given or not (binary decision), the systematic selection of test cases reduces the number of required tests from 2^n to $n + 1$. The procedure for causes interconnected by an "or" relation is similar.
When transferring this promising concept to M&S V&V (as indicated, but not completed by [Balci 1990]), simulation time introduces several complications. Although Cause-Effect Graphing allows it to express a great variety of behavior constraints (or Cause-Effect Dependencies), there is one serious drawback: It is assumed that any given combination of causes *immediately* (no expiration of time) results in one or more particular effects. This assumption is justified for the specification of (timeless) software functions or other elementary units, but not for the I/O behavior and the internal state changes of a symbolic model over time – here temporal dependencies are of great importance, including the causal order of states and state transitions. Thus, the only opportunity for the application of Cause-Effect Graphing in its unmodified, original form is during the V&V of the Executable Model, to aid the testing, whether the functions within the code of the Executable Model are built to their functional specification, given in the high level design. This is a pure software engineering task already discussed in [Myers 1979] and other publications. If one wants to transfer the idea of formal specification of behavior propositions to M&S V&V, one needs to change the foundations of Cause-Effect Graphing: A more expressive underlying mathematical logic is required.

5.3.2 The Enabler: Temporal Logic

A prerequisite to the verification of model behavior is to express the desired behavior of the model formally. As discussed in section 5.3.1, Boolean logic is not sufficiently expressive to formalize model behavior over time, as the causal order of model I/O and internal state changes is of great importance. Temporal logic seems suitable for the specification of desired model behavior, and showed to perform well in the context of model checking (see section 5.3.7).
To formalize desired behavior of a symbolic model *over time*, several sub-classes of temporal logic exist, including Propositional Linear Temporal Logic (PLTL) with or without "past" operators, Computation Tree Logic (CTL), CTL* [Katoen 1999], and Real Time TL [Ruf and Kropf 2001]. For the specification of behavior constraints already the least expressive PLTL

provides powerful features, which will be introduced briefly in the following. This introduction is only intended to provide the foundation for the approach taken in the following; a more fundamental introduction into Temporal logic can be found in, e.g., [Katoen 1999; Gerth, Vardi and Wolper 1995].
PLTL is a propositional logic with linear time, which means that alternative future developments are not allowed (there is only one "line of time"). PLTL supports the specification of safety and aliveness properties. The basis of PLTL are atomic propositions $p \in AP$ (the set of atomic propositions) that contain atomic statements that can be *true* or *false*. (For example, the atomic proposition $p_{infrontof(i,k)}$ may state that vehicle *i* drives in front of vehicle *k*, which can be true or false.) Those atomic propositions are combined to formulas Φ, according to the syntax of PLTL, which is given below in Backus-Naur Form. With $p \in AP$, the set of PLTL formulas is defined as

$$\Phi ::= p \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathbf{X}\Phi \mid \Phi \mathbf{U} \Phi .$$

For the explanation of the temporal operators, first the "time line", which is a triple *M=(S, x, Label)* is introduced, where

- *S* is a non-empty denumerable set of states,
- *x* is an sequence of states $s \in S$, and
- *Label*: $S \rightarrow S^{AP}$ a function, which assigns to each state $s \in S$ the atomic propositions *Label (s)* that are valid in *s*.

Or, in other words, the "time line" is a sequence of states in each of which particular propositions are true or false. In PLTL, time serves as a linear ordering relation on a sequence of states, it does not allow statements about the length of the time interval, which passes between two individual states.
The first three formulas yield the set of formulas of propositional logic. The temporal operator **X** is pronounced "neXt", and **U** "Until". The Boolean operators ∧ (conjunction), ⇒ (implication), ⇔ (equivalence), *true*, and *false*, and the temporal operators **F** ("Future") and **G** ("Globally") are derived from the above elements.

- If s_i is the current state in sequence *x*, then **X***p* means that *p* holds in the next state s_{i+1},
- *p***U***q* means that *p* holds in all following states of sequence *x* until the state s_j (with $j>i$), where *q* holds,
- **F***p* means that there will be a future state s_j in sequence *x* (with $j>i$) where *p* holds, and
- **G***p* means that *p* globally holds for all following states of sequence *x*.

Example: With AP = {p_{red}, p_{empty}}where p_{red} states "the lights are red", and p_{empty} states "there is no vehicle right beyond the lights", the condition "whenever the lights are red and there is no vehicle right beyond the lights, then this will not change as long as the lights remain red" can be formalized in PLTL as

$$\mathbf{G}(p_{red} \wedge p_{empty} \Rightarrow p_{empty} \mathbf{U} \neg p_{red}).$$

In the V&V Triangle, the concept of Cause-Effect Graphing in combination is used with the option to formally specify desired model behavior over time for M&S V&V.

5.3.3 Planning with the V&V Triangle

According to the V&V Triangle, already in sub-phase 1.1 temporal logics can be used to express documented behavior requirements formally, to demonstrate the unambiguousness of the Structured Problem Description, and to support assessment of its "correctness in fact" (consistency with the available domain knowledge). Then in V&V sub-phase 1.2, the explic-

itly formalized behavior requirements can be presented to the sponsor for feedback on the question of "completeness in content".
After system analysis, with the Conceptual Model available, the potentials of formal behavior specification can be explored in depth. In V&V sub-phase 2.1, for each conceptual submodel behavior propositions can be defined, based on the submodels input, output, and state parameters. After these behavior propositions were used to demonstrate that the submodel behaviors are "correct in fact", the behavior propositions are matched to the Structured Problem Description in V&V sub-phase 2.2, revealing errors or loopholes in both the Structured Problem Description and the Conceptual Model. Subsequently in V&V sub-phase 2.3, the behavior propositions can be discussed with the sponsor, to confirm that no important behavior propositions were forgotten. In V&V sub-phase 3.2, the behavior propositions are the foundation for model checking. By reusing them as specification of desired behavior, it is demonstrated that the Formal Model with a well-defined state space is "complete in content" and "further abstraction and idealization is suitable and correct". In the V&V sub-phase 4.3 test cases are generated that are most likely to cover the violations of the behavior constraints, if there are any. During both V&V sub-phases 4.3 and 5.4, all I/S/O-traces generated during testing and experimentation are checked for any violation of the defined behavior propositions.
For the M&S sample project, the identification of behavior constraints and the detection of their violation contributes to the following V&V plan, with the selection of V&V sub-phases that are processed visualized in Figure 18:

- Sub-phase 2.1 (Conceptual Model):
  - Demonstrate self-consistency, demonstrate unambiguousness: Create an initial list of formalized behavior propositions. Identify sections of the Conceptual Model documentation where the information required for this purpose is missing or ambiguous.
  - Demonstrate factual correctness: Present initial ("re-translated", if required) list of formalized behavior propositions to domain experts for review.

  The sample implementation is documented in section 5.3.4.
- Sub-phase 2.2 (Structured Problem Description-Conceptual Model transformation):
  - Demonstrate that the anticipated model behavior meets the requirements: Assign behavior propositions to behavior information within the Structured Problem Description, identify gaps in the behavior propositions list and the Structured Problem Description, adjust behavior proposition list as required, and document loopholes in the Structured Problem Description.

  The sample implementation is documented in section 5.3.5.
- Sub-phase 2.3 (Sponsor Needs-Conceptual Model transformation):
  - Demonstrate that the model behavior allows the needed experimentation: Discuss behavior propositions with the sponsor, adjust behavior propositions as required.

  The sample implementation is documented in section 5.3.6.
- Sub-phase 3.2 (Conceptual Model-Formal Model transformation):
  - Demonstrate that the Formal Model completely and consistently reflects the symbolic behavior description of the Conceptual Model: Use behavior propositions for (submodel-wise) Model Checking.

  The sample implementation is documented in section 5.3.7.
- Sub-phase 4.3 (Conceptual Model-Executable Model transformation):
  - Confirm that the Executable Model behavior is consistent with domain knowledge: Use behavior propositions for test case creation, generate I/S/O trace acceptance automata for I/S/O trace acceptance checking.

  The sample implementation is documented in section 5.3.8.
- Sub-phase 5.4 (Simulation Results-Executable Model transformation):

- Confirm that the Executable Model behaves during all experiments as intended with the Conceptual Model: Use I/S/O trace acceptance automata for post-experimentation analysis of generated I/S/O traces.

The sample implementation is documented in section 5.3.9.

The path is “closed”, allowing to backtrack the Simulation Results to the Sponsor Needs. However, it becomes obvious immediately, which sub-phases have not been addressed at all and might need coverage by other paths, as discussed in section 5.3.10.

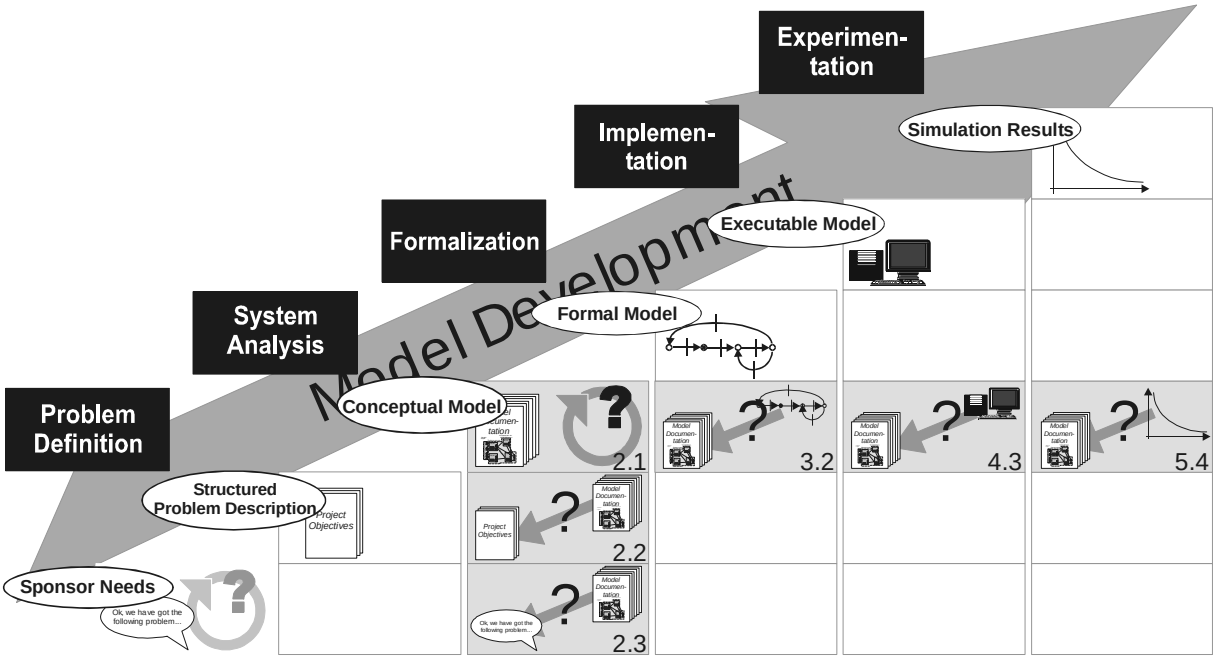


Figure 18: Path of behavior propositions violation detection

5.3.4 Sub-Phase 2.1: Extracting Behavior Propositions from the Conceptual Model

To achieve the V&V objectives defined for sub-phase 2.1, behavior propositions are extracted from the Conceptual Model. If there are internal inconsistencies or ambiguities in the Conceptual Model, they most probably result in inconsistent logical expressions, or are revealed during the (failed) formalization attempt. This effect is similar as in [Myers 1979], when he states that the accurate study of a natural language specification for the purpose of formalization reveals previously undetected errors.

The identification of behavior propositions in the Conceptual Model is closely related to the creation of a Cause-Effect Graph, as described in section 5.3.1. Both domain expert and modeling expert work together to accomplish this task.

1. Identify all input, state, and output parameters for each submodel. If the Conceptual Model is available as a data base entry according to a known template, this can be done automatically.
2. Carefully read the behavior description of each conceptual submodel and write down any implicitly or explicitly stated behavior proposition. Link the extracted behavior propositions to their origin in the Conceptual Model documentation. If missing parameters or other inconsistencies or ambiguities are detected, this shall be recorded.
3. Formalize the behavior propositions in PLTL, if possible.

Next to the detection of incompleteness or ambiguity, a subsequent review of the extracted behavior propositions by a domain expert may reveal inconsistency with domain knowledge.
Already the process of extracting the parameters of each submodel revealed serious deficiencies in the documentation of the Conceptual Model behavior, which was created using the template introduced in section 3.2, but without a rigorous control by, e.g., an electronic form with controlled entry fields. It was revealed that the description of the vehicles behavior was incomplete; no information about destination selection and route planning is given at all. However, although it was possible to express several important behavior propositions, numerous others could not be formalized in PLTL. Logics with greater expressiveness seem more suitable for the formalization of behavior constraints, which should be subjected to exploitation during future research (see section 6.10).

5.3.5 Sub-Phase 2.2: Counterchecking with the Structured Problem Description

To achieve the V&V objectives defined for sub-phase 2.2, the initial list of formalized behavior propositions from 2.1 is compared to the behavior requirements implicitly or explicitly stated in the Structured Problem Description. The aim with the highest priority is, to ensure that no behavior constraints identified as relevant in the Structured Problem Description are not reflected by the Conceptual Model. As a side-effect, also missing behavior requirements in the Structured Problem Description are revealed.
The V&V activities of this phase become even more important, if no Conceptual Model is available, which in the defense community is usually the case for legacy models (e.g., [Scholten 1998]). Then the list of formalized behavior propositions extracted from the Conceptual Model in 2.1 is replaced by a list of anticipated or assumed behavior constraints, which is created by "playing" with the legacy model and making assumptions about its underlying Conceptual Model.
The comparison between the Conceptual Model behavior propositions and the required behavior comparison is a highly specialized approach to a consistency and completeness check, which focuses on the comparison between required model behavior and conceptually described model behavior.

1. Assign the behavior propositions extracted from the Conceptual Model to the M&S objectives and the explicitly stated behavior requirements of the Structured Problem Description.
2. Identify M&S objectives and behavior requirements in the Structured Problem Description without behavior propositions assigned to; identify behavior propositions without an originating aim or a behavior requirement.
3. Analyze these gaps, trace them back to missing requirements in the Structured Problem Description, or missing behavior description in the Conceptual Model.

In the M&S sample project, the extraction of behavior propositions from the Conceptual Model resulted in numerous behavior propositions without having a counterpart in the Structured Problem Description. As these behavior propositions were consistent with the M&S objectives, there was no need to modify the Conceptual Model. Theoretically an extension or refinement of the Structured Problem Description was possible, but because no inconsistencies were found, direct feedback with the sponsor is preferable.

5.3.6 Sub-Phase 2.3: Counterchecking with the Sponsor Needs

To achieve the V&V objectives defined for sub-phase 2.3, the formalized behavior propositions are presented to the sponsor for review. If required, they are translated back into a language that the sponsor understands. From the discussion of the behavior propositions extracted from the Conceptual Model the sponsor gets another indicator for the suitability of the model for the intended purpose.

This approach to the validation of the Conceptual Model behavior description benefits from the diversity of information representation. The sponsor with the real counterpart of the model use in mind can quickly realize, which behavior propositions are reasonable, and which are not.
For behavior propositions review, the sponsor or a sponsor representative with knowledge of the higher level M&S aim focuses the Conceptual Model review activities on the explicitly made available Conceptual Model behavior description without distraction by any other Conceptual Model contents.

1. Translate the list of formalized behavior constraints back into natural language.
2. Discuss the list with the sponsor.

For the M&S sample project, in the list of behavior propositions both the PLTL formula and its natural language translation are available. A meeting with professional analysts for in-town road traffic took place, were several of the behavior propositions extracted from the vehicles Conceptual Model were discussed. The analysts doubted that the (completely deterministic) vehicles behavior unambiguously described by the behavior propositions is suited to reflect the often irrational behavior of real vehicles.

5.3.7 Sub-Phase 3.2: Model Checking

To put it in a nutshell, model checking is an automated technique that, given a finite state model of a system and a property stated in some suitable logical formalism, systematically checks the validity of this property. Examples of successful model checking projects can be found in [Kim, Park, and Baik 2000; Hong and Kim 1996; Geilen 2000; Bienmüller et. al. 2001]. In the following the applicability of model checking for the proof that the Formal Model satisfies the desired behavior propositions is discussed.

5.3.7.1 Automaton Creation and Acceptance Checking

To verify that a finite state model satisfies some formally specified behavior propositions, these propositions can be transformed into acceptance automata, according to a well-defined algorithm, which is documented in, e.g., [Katoen 1999; Gerth, Vardi and Wolper 1995]. Acceptance automata only accept input sequences, which satisfy the behavior propositions defined by their underlying PLTL formula. Such an automaton will be used to check, whether the (state transition) behavior of a model is accepted (and correct) or rejected (which implies a violation of the PLTL behavior specification). Figure 19 depicts the acceptance automaton $A_{stop\_at\_lights}$, which was generated from the formula $\mathbf{G}(p_{red} \wedge p_{empty} \Rightarrow p_{empty}\, \mathbf{U}\, \neg p_{red})$, identified during the M&S sample project in sub-phase 2.1. This acceptance automaton is a finite state machine with one or more entry states (incoming arrows without origin) and one or more accepting states (double outline). The unidirectional arrows depict all allowed state transitions. Each state of the acceptance automaton is labeled, indicating which input parameter values (propositions) are accepted in the particular state. (Elements of the set of atomic propositions that are not contained in the label of a state are assumed not to hold in the particular state.)
With an acceptance automaton and a formally specified state transition model available, two approaches of using it are popular:

- The state space of the model is traversed, ensuring that each state has been visited at least once. For each state transition generated this way it is checked, whether there is also a transition in the acceptance automaton, which leads to a state, which accepts the properties of the successor state in the model. Non-acceptance implies the violation of the behavior constraint.
- The state transition model is considered as an automaton describing the possible model behavior. The PLTL formula specifying the desired behavior is negated prior to its transformation into an acceptance automaton, which yields an automaton only accepting *unde-*

sired behavior. The product automaton of both automata (acceptance automaton × state transition model) should not accept any behavior, else a violation of the behavior constraint is detected [Katoen 1999].

It was shown that non-trivial models that are precisely specified using the DEVS formalism can be exhaustively verified in the way described above [Hong and Kim 1996]. In the domain of Integrated Circuit Design huge (but more or less homogenous) models are build. Here, model checking is well established for the detection of violation of behavior specifications [Burch et. al. 1994].

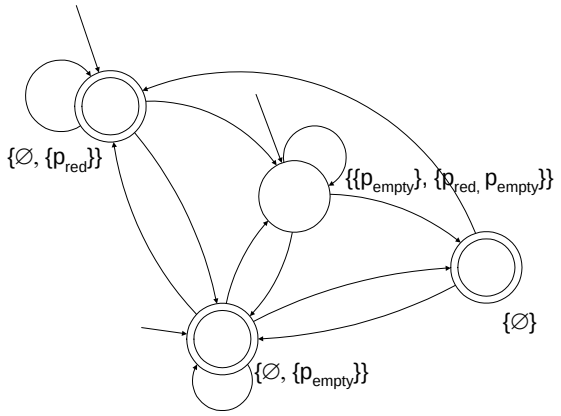


Figure 19: Acceptance automaton $A_{stop\_at\_lights}$

5.3.7.2 Comments on Implementation

Although in the M&S sample project a Formal Model specified according to the DEVS formalism is available (0), it would have mend serious additional effort to refine it to a complete state transition model that allows model checking as proposed in [Hong and Kim 1996]. Also the extraction of a flattened state space from the available Formal Model, which may serve as input to a specialized model checker as the Symbolic Model Verifier SMV [SMV 2.5 1998], would have meant a tremendous additional effort. However, it must be assumed that the necessary additional effort can be significantly reduced, if the development of the Formal Model is computer aided, yielding a machine-readable specification, free of syntactical and semantic errors, which prohibits "workarounds" as found in the available DEVS specification. To find acceptance in the M&S-Community, these tools also should provide strong support for the subsequent implementation. To simplify the specification of DEVS models and to open DEVS to a wider user community, considerations about graphical visualization of DEVS can be found in [Brade and Cegla 2003]. For an in-depth examination of this issue, a diploma thesis was started [Schäfer 2003].
The complete absence of an Formal Model is not unusual, and the V&V objective "Demonstrate completeness in content" of sub-phase 3.2 is hardly achievable. In this M&S sample project, the V&V objective "confirm completeness in content" of sub-phase 4.3 changed to a "demonstrate", too.

5.3.8 Sub-Phase 4.3: Automated I/S/O Trace Analysis

With the Executable Model (simulation software) available, it becomes possible to actually run the model for testing, and to analyze its behavior. There is a difference between *software testing* one the one hand, and *model testing* on the other, although the border is blurred (see also section 1.5.3). The author considers everything that deals with the detection end elimination of bugs (e.g., program crashes, memory leaks, or user interface malfunction) in the simu-

lation software as purely software related and out scope of this document. However, when one concentrates on model testing, one is also likely to detect coding errors that remained unrevealed during software quality assurance.

5.3.8.1 I/S/O Trace Generation and Acceptance Checking

The structure created from all input, state, and output variables of an Executable Model is called the *I/S/O vector* in the following, where an I/S/O vector resembles a state $s \in S$ as defined in section 5.3.2. Then, the *I/S/O trace* documents the sequence x of I/S/O vectors, i.e., all input, state, and output values that were created during an execution of the Executable Model, ordered by their chronological occurrence. (Time stamps for the I/S/O vectors are helpful, if, e.g., the I/S/O trace is visualized or compared to other log files, but are not required for the detection of violations of behavior constraints specified in PLTL.)
As discussed in section 4.5.1, there are two aspects on testing a model:

- the development of test cases, and
- the analysis of the model behavior observed during the test.

For the development of test cases, one uses the modified list of behavior propositions to derive model initialization conditions that are most likely to provoke a model state in which the condition is violated. The model behavior during the test is recorded as an I/S/O trace, which requires all state variables of interest made visible by instrumentation. Then the acceptance automata created from the behavior propositions are used to automatically analyze the I/S/O traces.
Following exactly the same procedure as for model checking, from the PLTL specification of a desired behavior proposition, an acceptance automaton is generated. Now – in contrast to model checking – an input to the acceptance automaton consists of a subset of values of the I/S/O vector, while the complete input sequence is provided by the I/S/O trace. Whenever a new input is provided, a state transition in the acceptance automaton is attempted. An I/S/O trace is only accepted, if for each contained next I/S/O vector, there is a suitably labeled successor state in the acceptance automaton that can be reached.

5.3.8.2 Comments on Implementation

The main difference between model checking and automated I/S/O trace analysis is that for the trace analysis the claim for exhaustiveness of checking is omitted. While model checking concentrates on proving that no possible state transition violates the formally specified behavior, during trace analysis only a subset of all possible paths through the state space is evaluated. As for any testing technique, the identification of high-yield test cases and the monitoring of the current internal state are crucial. With the source code available, for the sample evaluation it was feasible to instrument the Executable Model to log a suitable I/S/O trajectory.
Generating an acceptance automaton from a PLTL specification of the behavior constraints, and feeding an I/S/O trace into the acceptance automaton can be automated. The tool VIOLADE (VILOLAtion DEtector) was designed in the context of this thesis and prototypically implemented, based on the approach described above. The user specifies the format of the I/S/O trace (trace file format), specifies a desired property in PLTL, identifies the I/S/O trace file that is subjected to examination, hits the run button, and gets the answer, where in the trace file the specified behavior proposition is violated, if there is any violation. More details concerning VIOLADE are found in [Brade and Waldner 2003].

5.3.9 Sub-Phase 5.4: I/S/O Trace Surveillance

During experimentation, the same I/S/O trace acceptance checker can be reused, which was created for analysis of the I/S/O traces during testing the Executable Model. Also during ex-

perimentation, undesired model behavior may be revealed that was not previously detected during Executable Model testing. The procedure is exactly the same as the second part of the procedure for Executable Model testing, and with a violation detection tool as VIOLADE available, the additional effort is small.

5.3.10 Concluding Remarks on PLTL for Behavior Propositions Specification

The advantages of the formal specification of behavior constraints in PLTL include, that the natural language specification of the model behavior and the behavior requirements in the Structured Problem Description are intensively reviewed, and that violations of the formally specified behavior propositions can be reliably detected even within extremely large samples of I/S/O traces.

The limitations are serious; although important constraints can be formally expressed in PLTL, numerous other constrains remain that cannot be formalized due to the limited expressiveness of PLTL, and therefore their violation is not automatically detected. Also, undesired behavior that was not anticipated and expressed as a behavior proposition, but becomes obvious to an attentive human observer will never be detected by an acceptance automaton (see also section 5.4). Thus, at the current state of the art, automatic violation detection of behavior propositions cannot replace other methods for assessing the correctness and suitability of model behavior, but only complement them. More expressive logics, and the generation of more powerful acceptance automata need to be explored for improved acceptance checking (see section 6.10).

5.4 Human Review

Although automated computer-based evaluation techniques are more objective, more efficient, more likely to be repeatable, and even more reliable than human review, the human reviewer plays an extremely important role for the V&V of models and simulation results. This statement is based on the following observations:

- Today, M&S still are considered as both art and science. Models are build in numerous application domains, which sometimes lack formal specification techniques for the problems within this domain. The lack of discipline, or the disability to rigorously specify requirements prohibit the use of tool support for the evaluation of the requirements specification, and require a high degree of human involvement.
- When specifying the requirements for a model, humans make numerous implicit assumptions, which are not all explicitly stated or formalized. The violation of these implicit assumptions can only be detected by an expert observer, who knows about them.
- Models and simulation results are often an integral part of an iterative knowledge gaining process. The Executable Model and the observation of Executable Model behavior can serve as stimulus for the human mind. Until completing the Executable Model, there may only be a vague expectation of the model behavior, which can be only be *judged for plausibility* by its observation.
- For validation, there remains a gap between the real world and the M&S world, as today no description of the real world can be proven to be suitable (although unsuitable descriptions sooner or later are identified as being unsuitable). Therefore, the assessment of validity of a model or simulation results for an intended purpose remains to the human.

Within this section only those review activities are considered as human review that directly deal with the intermediate products or interpreted behavior data (provided in a human-readable form), but not with the evaluation of V&V results (gained by, e.g., static analysis of the intermediate products; for example, the review of a control flow graph yielded by control

flow analysis is out of scope of this section. Also any tracing problems that require a high degree of specialized human involvement are addressed in section 5.1).

5.4.1 Background – Current Practice

Human review currently is *the* method not only in the defense community for determination of model validity (at least for constructive simulations), thus, there was no transfer of an idea from another research area required (as for documentation correctness in form checking or the detection of behavior propositions violation). Human review helps to reveal many inadequacies of models, as will be made plausible in the following, but it must not be overestimated. Human review and more objective approaches to V&V are complementary, which means that one never should rely on human review or automated error detection alone.

5.4.2 Enabler: Human Knowledge and Flexibility

A human features a high robustness to unknown document formats, and can access broad background knowledge for the detection of errors, which are not directly in the scope of an explicit search. The intermediate product can be provided in (nearly) any format, and a list of review criteria does not need to be developed into detail (if required at all), as the human mind can adapt itself to the problem it is confronted with. However, the reliability of the review results depends on the skill, personal motivation, bias, concentration and other hardly quantifiable human factors, which makes them hardly repeatable, if the expert changes. Even if the same expert is confronted twice with the same review task, it must be assumed that there are discrepancies between the review results.
When involved in the V&V of a model or simulation results, there are three main tasks for the human reviewer:

1. Review the documentation of the intermediate products (static analysis), to assess its suitability;
2. "manually" or "mentally" execute the symbolic behavior specification provided in the Conceptual Model (which requires more interpretation than a machine is capable of);
3. review execution results (dynamic analysis), to asses their suitability.

Ad (1): The intermediate product is passed on to experts knowledgeable in the domain of interest who carefully analyze the document, using their expert knowledge or checklists. Well known review techniques from Software Engineering that can be transferred directly to M&S include *Peer Review*, *Audit*, *Desk Checking*, *Self-Inspections*, *(Formal) Inspections*, or *Walkthroughs*, as explained in detail in [Smith and Wood 1987; Beizer 1990; DeMillo et al. 1987; Hetzel 1984; Myers 1979; Office of Safety and Mission Assurance 1993]. (However, the checklists need to be adjusted to ensure the coverage of all aspects relevant for M&S.) Although they are not based on a mathematical foundation and tool support is low, these techniques contribute significantly to quality assurance in Software Engineering, and are likely to have a similar effect on M&S. It was observed that, if groups are involved for review, it is often the same person who created the code who also detects the faults in the code, simply by being encouraged to present it deliberately to others [Office of Safety and Mission Assurance 1993].

Ad (2): If the intermediate product lacks the required unambiguousness for automatic interpretation, or, if there is no automated interpreter available, a human may manually or mentally interpret the behavior specification. Then from the mentally interpreted behavior conclusions are drawn back to the behavior specification.

Ad (3): As soon as there is a behavior specification that may be interpreted by a machine for automatic generation of behavior data (Executable Model), these execution results are reviewed by human experts. This approach is highly popular in the M&S community. If the experts assess the execution results giving consideration to the intended purpose directly

as suitable or not, this process is called *Face Validation*. It has several serious drawbacks, if used exclusively for validation (see section 3.1.4), but it is of great value in combination with other more objective techniques. An attempt to work around the reviewers bias is the *Turing Test*, where the reviewers do not know the origin of different data sets presented.

Tool support for human review is limited to test case generation, visualization and animation, data representation, expert guidance, and expert opinion recording tools, as it lacks the required formal foundation for the creation of tools with a higher degree of automation. It is imaginable that the intermediate product is fed into a suitable expert system, which uses a domain-specific knowledge base for the evaluation of the contents, to replace the human reviewer. This requires machine-readable documentation of the intermediate product and a suitable knowledge base. However, until today expert systems showed only to perform well in a few, highly specialized application domains, and the tasks introduced above remain to the human being.

5.4.3 Planning with the V&V Triangle

Due to the exploratory nature of the M&S sample project, there was no real sponsor with real Sponsor Needs. This prohibited the demonstration of completeness and consistency with the Sponsor Needs (Sub-Phases 1.2, 2.3, 3.4, 4.5, and 5.6), which would be essential in a real V&V project. The domain knowledge required for expert review in the M&S sample project concentrates on performance analysis of road traffic flow. In the V&V Triangle no human review is planed for the Formal Model (sub-phases 4.x). Figure 20 depicts the sub-phases of the V&V Triangle that are touched with the following V&V plan:

- Sub-phase 1.1 (Structured Problem Description)
  - Demonstrate consistency with domain knowledge: Review parameter names, dimension unit, accuracy, and the description of anticipated submodels, their attributes (interfaces), and submodel interactions.
  - Demonstrate self-consistency: Review I/O parameters and experimentation requirements, and demonstrate the suitability of the overall model I/O for the planed experiments.
- Sub-phase 2.1 (Conceptual Model)
  - Demonstrate consistency with domain knowledge: Review the parameter names, their dimension units, and accuracy; review submodels, their attributes (interfaces), and submodel interactions; mentally execute the submodel behavior specifications, guess behavior of higher level submodels; compare guessed behavior to domain knowledge, reference knowledge sources.
  - Demonstrate self-consistency: Review the attributes of the submodels and the specifications of their interactions, demonstrate the suitability of the interfaces; review the attributes of the submodels and their next higher level submodel; demonstrate suitability of composition or aggregation.
- Sub-phase 2.2 (Conceptual Model)
  - Demonstrate completeness and consistency with the Structured Problem Description: Review model border, structure specification, and behavior specification, giving consideration to required I/O parameters, accuracy, and domain knowledge.
- Sub-Phase 4.1 (Executable Model)
  - Demonstrate consistency with domain knowledge: Graphically visualize and animate the (executable) model behavior; compare observed model behavior and submodels behaviors to domain knowledge; record results of comparison.
- Sub-Phase 4.3 (Executable Model)

- Confirm consistency with the Conceptual Model: Graphically visualize and animate the (executable) model behavior; compare observed model behavior and behavior described in the Conceptual Model; record results of comparison.

- Sub-Phase 4.4 (Executable Model)
  - Confirm consistency with the Structured Problem Description: Graphically visualize and animate the (executable) model behavior, compare observed model behavior and behavior requirements defined in the Structured Problem Description; record results of comparison.
- Sub-Phase 5.1 (Simulation Results)
  - Demonstrate consistency with domain knowledge: Compare simulation experiment design with design of (fictive) real experiment; assess Simulation Results for plausibility.
- Sub-Phase 5.4 (Simulation Results)
  - Confirm consistency with the Conceptual Model: Explain Simulation Results based on the assumptions made for the Conceptual Model.

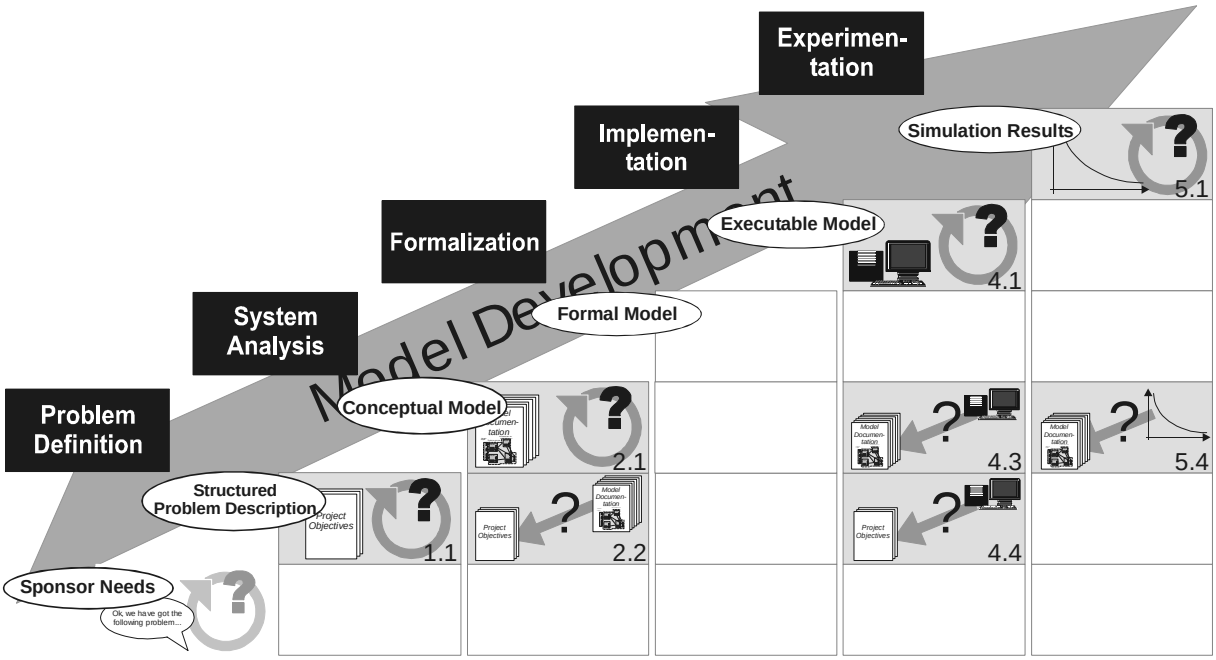


Figure 20: The path of human review

5.4.4 Sub-Phase 1.1: Problem Description Review

The human reviewer with knowledge of the application domain and M&S, but not necessarily of the Sponsor Needs, assesses the correctness of the Structured Problem Description with respect to his domain knowledge. The choice of I/O parameters for the M&S sample project was found to be acceptable for performance analysis. This was confirmed during a meeting with experts from the Verkehrsordnungsamt (Traffic Regulation Office) München.

5.4.5 Sub-Phase 2.1 and 2.2: Symbolic Description Review and Mental Execution

The symbolic description of structure and behavior of each submodel is reviewed for correctness with respect to the reviewer's domain knowledge as a participant in daily road traffic, and expert knowledge taken from [Topp 1994; Lemessi 2001; Tittelbach 2001]. This is performed independently from the intended purpose of the model development and use, but under explicit consideration of the references used for conceptual modeling (documented in Appendix A), and additional available domain knowledge. Each submodel was individually

mentally executed to determine, whether it behaves as desired and consistently with the available domain knowledge. Whenever possible, conclusions for the submodel interactions were drawn, and from these behavior assumptions for the next higher submodel were made. It was also checked that the model behavior anticipated by mental execution allows the implementation of the planned experiments as specified in the Structured Problem Description.
The activities in this sub-phase directly impacted model development, as they can hardly be separated from it, as long as reviewer and developer are the same person. Whenever a new behavior description was completed, it was immediately checked by its mental execution, whether it satisfies the modeler's intention. This activity is essential, and the effort invested in review and mental execution directly paid back for the following developmental activities. During the M&S sample project, the review of the symbolic structure description revealed that there was a misinterpretation of the CAD sketch of the intersection layout, which resulted in a wrong assignment between a lights node "l51" and its controlling lights group. It was also detected that the assumption of an exponential distribution of the vehicles inter arrival times is inconsistent with the domain knowledge documented in [Topp 1994].
However, to create a meaningful V&V product from Conceptual Model review and mental execution other than a review report is difficult. The credibility of this report directly depends on the credibility of the expert who wrote it. To perform V&V as efficiently as possible, the reviewer should not change for sub-phase 2.2. Although errors in the Conceptual Model can be detected efficiently by symbolic description review and mental execution, it must be assumed that other significant errors remain undetected at this stage of model development due to the complexity and unpredictability of the submodel interactions.

5.4.6 Sub-Phase 4.1, 4.3, and 4.4: Visualization by Animation

The huge amount of data created during model execution in endless columns of numbers is not intuitively comprehensible for a human being reviewing it. Thus, unsuitable or incorrect model behavior data may pass data review undetected. To allow a more intuitive assessment of model behavior, the model behavior data is visualized over time, which means that time is not only just another dimension in a function graph, but the dynamic change of parameter values can be observed at the computer screen.
For validation of the Executable Model, test cases should be created that are most likely to reveal undesired model behavior (high yield test cases). Expert opinion plays an important role for test case generation as well as for model output analysis. A single test or the multiple execution of one type of test already grants some insight into the behavior of the Executable Model, but the implementation of a *suite of tests* is more likely to reveal hidden errors.
In implementing visualization and animation for V&V in Sub-Phase 4.1 using Wolverine's Proof Animation [Wolverine Software 2003], the animation provided a bird's perspective on the intersection. The following test techniques on the path of human review were inherited from Software Engineering and performed on the M&S sample project:

- *Random testing*: Animation of the trace of an arbitrary simulation experiment quickly revealed that
  - there was an erroneous link in the intersection graph. Vehicles "jumped" around the corner after passing an "intersect node";
  - the idealization of vehicle width to zero moves the vehicles closer to intersect nodes and join nodes than physically possible;
  - the left-turning behavior of the vehicles is too bold; they enter the opposite side lane, if there is no vehicle closing in on the lane that is close to the driver, and cause (undetected) crashes with new incoming vehicles when delayed by a vehicle on a more distant opposite side lane.
- *Stress testing*: Originally developed to detect memory leaks and problems with the storage and access of large amounts of data [Myers 1979], the modified stress test fea-

tures the generation of a vast amount of (simulated) objects. This is not performed for the purpose to test software quality (however, detection of problems in dealing with software objects may be a side effect), but to test, whether the concept of object interactions is still correct for large numbers of objects. In the M&S sample project for stress testing the inter arrival times in all vehicle sources were reduced to ten seconds, quickly creating traffic jams of considerable length. Animation revealed that vehicles generated for entry roads without remaining capacity (a long vehicle queue was already waiting for green lights) entered the road as if it were empty. After the detection of this error it could easily be traced to a fault in the predecessor determination function of the vehicle.

- *Symptom generation test*: An expert uses his knowledge to make up situations (input conditions and expected output) he believes to be symptomatic for model validity that the model shall reproduce. After (or during) running the model it is determined, whether the actually observed model output is sufficiently similarly to the desired model output. It was tried to provoke a traffic jam on the left turn lane coming in from north by creating extremely dense straight traffic from the south. The queue definitely grew to slow, as the extremely bold left turning behavior of the vehicles lead to an unrealistic queue length reduction.
- *Degeneration test*: This is actually a modified symptom generation test. The model is not executed as it is, but parts of its functionality or submodels are disabled or removed. After that, an expert defines some desired behavior of the modified model, derives the conditions that must be reflected by model input to result in this behavior, and tests subsequently, whether the observed behavior matches the desired behavior. The degeneration test revealed that the behavior rules of the vehicles do not reflect the driving behavior suitably – best intersection performance data (throughput, mean vehicle waiting times) are generated when the lights are completely removed. The vehicles behave extremely deliberate and conscious, decelerating only, if necessary, and accelerating speedily. Crash situations created by this behavior are not detected in the simulation.
- *Traces driven methods*: For this test, the test setup is driven by the available traces of real measured data. Ideally measured output of the real system is also available, allowing subsequent comparison between observed model behavior and recorded behavior of the real system. Unfortunately this did not provide any additional insight, as the available data was to inaccurate (vehicles/minute) to allow the creation of another input trace as already used by the model internal vehicle stream generator.
- *Sensitivity Analysis*: The vector or trace of input values was changed slightly from simulation run to simulation run to allow the determination of the impact of each input parameter. As there are usually too many possible modifications of the values of the input vector or trace to test them all, expert knowledge is required to identify high-yield sets of input vectors or traces. In subsequently evaluating test results the expert compares the observed impact to his knowledge.

Systematic stimulation of the Executable Model is necessary for testing, but not sufficient. On the path of human review, the model output is graphically preprocessed to support its judgment by humans. Function graphs were not applied in the M&S sample project, but the object flow was visualized and animated over time. It was presented to and discussed with experts from the Verkehrsordnungsamt München who instantly discovered anomalies in the vehicles behavior.

5.4.7 Sub-Phase 5.1: Results Assessment

During this sub-phase expert opinion is required to assess the adequacy of the experiment setup. For the M&S sample project, an experiment with 81 simulation runs was conducted with all permutations of three different densities over the three incoming roads on three different lights control tables. This was found to be a reasonable approach to general traffic flow examination by the experts from the Verkehrsordnungsamt München.

5.4.8 Sub-Phase 5.4: Requirements Satisfaction Assessment

It is not sufficient, if observed model behavior seems plausible to the expert; it is also necessary that it can be explained with respect to the abstraction and idealization that was applied to the real world and is documented in the Conceptual Model. With a M&S sample project of limited complexity it was easily possible to trace any observed anomalies back to their origin in the Conceptual Model.

5.4.9 Concluding Remarks on Human Review

Although conducted informally, human review of the Conceptual Model, and visualization and animation of the behavior of the Executable Model helped to detect numerous errors in the M&S sample project. There are two indispensable prerequisites for successful human review:

1. The complete and comprehensive documentation of the Conceptual Model and of all of its submodels making the model transparent for the analyst is available.
2. There is an in-depth visualization of the behavior of the model and all of its submodels, giving the observer insight into the model and submodel dynamics.

The review of the Conceptual Model occurs in the early stages of model development and therefore is a valuable early error detection measure. To increase the unambiguousness in the Conceptual Model, it is desirable to increase the share of formal elements within the Conceptual Model, as long as its readability for those knowledgeable in the application domain is not reduced. This also enhances tool support (see section 6.10).
Visualization and animation increase transparency of the model behavior, and help to reveal at least frequently occurring erroneous model behavior. It is desirable to support visualization and animation of model behavior early in the model development process, which again requires stronger tool support for formal modeling. However, special care is required when using visualized simulation output for validation; visualization tools can distract the observer from the essential model output, and errors in model behavior can be hidden by nice animation features.

6 SUMMARY AND CONTINUING WORK

Although the process of model development received a lot of attention during the last few decades, integrated quality assurance methods and measures have played only a subordinate role. In earlier processes [Shannon 1975; Schmidt 1985], validation was a concluding activity after model development is completed. In other process models the execution of V&V activities is implicitly assumed to be a part of the developmental activities, resulting in feedback loops to previous phases. With the desire and demand to increase the credibility of models and simulation results by increasing their (perceived) correctness and suitability, the need was expressed for explicit guidance on how to plan, implement, and document V&V activities in detail.
This section summarizes the motivation for this thesis, the starting position including the identified deficiencies in M&S, the approach proposed by this thesis to their elimination, and the lessons learned from applying the proposed approach to a test case.

6.1 Contribution to the State of the Art

When this work was initiated, several distinct approaches to the V&V of a model or simulation results existed, all described at abstract levels [Defense Modeling and Simulation Office 1996 and 2000; Balci 1990; Sargent 1999; Pace 1999a]. More specialized approaches with a higher level of detail as, e.g., [Flight Mechanics Panel Working Group 12 1985] were rare. In this thesis the similarities and differences between these approaches were identified, and from the analysis results the generalized V&V process – the "V&V Triangle" – was developed as proposal for a basis of future V&V.
This document contributes to the current state of the art

- an **overview and analysis of the major approaches to V&V** of models and simulation results currently known, and of successful quality assurance measures from related areas;
- a **consistent framework for V&V** of models and simulation results, including terminology, a discussion of the impact of available knowledge of both the model and the real system on the overall V&V effort, and thoughts on the main drivers of V&V, which are the worst case consequences of the application of simulation results;
- a **structured three-dimensional approach for information and documentation collection** in form of intermediate products that are created during model development, to provide a stable foundation not only for the execution of V&V activities, but also for the reuse of submodels;
- an **overview of dependencies between related V&V objectives** in form of the **V&V Triangle**, and thereby provides guidance for adaptation of V&V activities to available information about the model and the real system (tailoring);
- detailed **guidance for the selection of V&V techniques**, which can be efficiently and effectively applied for achieving a particular V&V objective, and a refined explanation of the V&V techniques application;

- the **documentation of lessons learned** by the application of several V&V techniques to a sample M&S project, and a clear vision for future V&V research.

Major aspects of this contribution, including the overview over the dependencies among related V&V objectives and the detailed guidance for the selection of applicable V&V techniques, are contained in the "V&V Triangle" documented in Appendix A.

6.2 The V&V Triangle

The V&V Triangle approach was derived from rigorous analysis of existing approaches to V&V of models and simulation results, personal experience gained during model development efforts, and experience with the V&V of models developed at the institute. The explanation of the V&V Triangle contains tables and figures to keep the V&V Triangle as transparent as possible. The V&V Triangle supports a structured approach to M&S V&V:

- Its inputs from the model development process are **well-defined intermediate products** that are created during model development, following the chronological order in which they become available;
- the **intensity and rigor of V&V** are variable, according to the risk incident to the intended purpose of model use, supporting establishing credibility incrementally;
- all **V&V objectives** identified within the process are harmonized with each other;
- previously created **V&V evidence is reused** efficiently during subsequent V&V activities;
- its output is the **modular documentation** of the V&V objectives achieved, the V&V activities conducted, and the V&V results which can be used to inspire belief in the correctness and suitability according to the identified risk.

For this purpose the V&V Triangle provides:

- detailed **documentation requirements** for the form and content of the intermediate products (providing all the desired information about the model and the simulation results);
- a **summary of error types**, which are most likely to be hidden in the intermediate product on which the V&V activities are focused;
- **V&V main phases** to assess the correctness and suitability of each intermediate product individually;
- **V&V sub-phases** with generic V&V objectives to fully explore the potential of the examination of specific types of intermediate products and external information;
- guidance on how to apply **V&V techniques** to achieve the V&V objectives, and on how to reuse previously created V&V products;
- an overview of the **dependencies between different V&V objectives** and the influence of repetitive examination on the credibility of the model or simulation results.

The efficiency of the V&V Triangle was evaluated by its application on a small, but not trivial road intersection model (Appendix C), revealing potential benefits and limits.

6.3 Potential Benefit and Limits of the V&V Triangle

The V&V Triangle provides a foundation for the verification and validation of models and simulation results. It is no procedural approach for planning and implementing V&V activities; rather, it needs to be tailored to the particular needs of any given M&S project. Although it provides valuable guidance how to plan and implement V&V, there also are limitations to its use.

6.3.1 Potential Benefit

- The application of the V&V Triangle is **independent from the chosen model development paradigm**. The V&V objectives and techniques summarized in the V&V Triangle depend on the minimum information that needs to be provided by the intermediate products. As long as these requirements are met, the V&V Triangle provides guidance on how to accomplish the V&V objectives for the intermediate products developed utilizing any model development process.
- The V&V Triangle is a **solid foundation for V&V planning and implementation**. The requirements for documentation of the intermediate products, the V&V objectives, and their associated V&V techniques were derived from M&S theory, existing approaches to M&S V&V, discussion with V&V experts, and personal practical experience with V&V planning and implementation. The dependencies between the objectives and techniques are clearly identified, and at least for a subset, the documented sample application gives additional insight into the V&V of models and simulation results.
- The V&V Triangle is a **foundation for further research** and for a wide range of applications. With the identification of the influences on V&V and their qualitative discussion, the foundation for rigorous quantitative analysis of the problem domain is laid. Availability of information and knowledge about the real system and the model itself limit the maximum possible V&V. Ideally, the activities should be exclusively driven by the maximum acceptable risk for the use of the model or the simulation results.
- The V&V Triangle points out the **dependencies between the V&V objectives** and thereby supports deliberate tailoring: Depending on their assignment to their subphases, particular V&V activities can replace other activities. At the expense of rigor and intensity, V&V can become less resource intensive.
- The V&V Triangle provides the **foundation for the implementation of less abstract application domain-specific V&V process models**. Depending on the application domain, V&V objectives and techniques identified in the V&V Triangle can be refined or replaced flexibly by more appropriate objectives and techniques.

6.3.2 Limits

- Despite its high level of detail, the V&V Triangle does **not define a procedure** to be followed strictly. It provides an overview of desirable V&V objectives and possible techniques for the achievement of these objective, but does not replace a deeper understanding of the problems regarding M&S V&V.
- Among the foundations of the V&V Triangle there is the assumption of model transparency. Although testing and analysis techniques can (and should) be combined and used to complement each other, the V&V Triangle does **not provide explicit guidance on how to deal with "black boxes"** for which there is no documentation. If an intermediate product is missing, the V&V objectives in the sub-phases associated with this intermediate product can be shifted into the sub-phases to their right, but this impacts the dependencies between the V&V objectives in a non-trivial way.
- The **reuse** of parts of the intermediate products is **not explicitly addressed**. If such parts are reused, leveraging the outcomes of V&V (or V&V related tasks) previously applied offers great potential for the reduction of the remaining V&V effort.
- Although the hierarchical organization of models is stressed, the benefits of the **characteristics of component-based models and distributed simulations are still not fully explored**. The V&V Triangle needs adaptation to these extremely promising sub-disciplines of M&S.

6.4 Application V&V Techniques

In the context of this thesis, the purpose of choosing and applying a V&V technique is the creation of evidence that demonstrates that an intermediate product satisfies some required or desired property. (As a side effect, errors are detected, if the intermediate product does not satisfy these properties.) Some V&V techniques are quite cumbersome to apply without appropriate tools available, others provide deep insight into the model and its behavior with little effort. The groups of V&V techniques or techniques discussed below demonstrated to be of special importance during the evaluation of the V&V Triangle in the sample M&S project.

6.4.1 Requirements Tracing and Configuration Control

For the purpose of requirements tracing a trace from the contents of an intermediate product to the associated contents of its predecessor is created. This allows one to track all features of the Executable Model back to the requirements specification given in the Structured Problem Description. The detection of inconsistencies between the intermediate products is supported by this technique. However, without appropriate software for model documentation available, which automatically controls the modifications of each intermediate product, such inconsistencies are quickly introduced. The implementation of requirements tracing does only require an extremely limited amount of creativity, and becomes a cumbersome and error prone task, if performed manually.

The conclusion is that requirements tracing without an adequate model documentation system is extremely time consuming and should be replaced by more efficient techniques for the detection of inconsistencies. However, if such a system is used and requirements tracing can be automated or supported (at least to a certain degree), there could be truly consistent model documentation. The author considers the design and implementation of a highly specialized model documentation system to be a desirable goal (see section 6.10).

6.4.2 Automated Error Detection

During the evaluation phase of the V&V Triangle, Propositional Linear Temporal Logic (PLTL) was used to specify desired behavior properties. Similar to model checking, from these properties acceptance automata were derived that are able to automatically detect violations of the desired behavior in the I/S/O traces of the Executable Model. Among the advantages of this approach to error detection is the efficiency of the technique, as large amounts of model behavior data can be analyzed automatically, and there is (practically) no chance, that a violation remains unrevealed. However, another finding of the attempt to automatically detect behavior violations in the test case was that it was impossible to express all identified behavior properties in PLTL due to its limited expressiveness.

Given the current state of the art, automated error detection methods cannot replace human review. In the author's opinion both approaches to V&V should be used to complement each other. The absence of errors that can be automatically detected then is proven during an automated procedure, while it remains to the human expert to detect more subtle violations of desired behavior.

6.4.3 Instrumentation, Visualization, and Face Validation

Although applicable only late in the model development process (an Executable Model is required that actually transforms input data to output data), visualization of the submodel behavior and their interactions, combined with face validation by SMEs was recognized as an extremely powerful approach to revealing numerous propagated errors and inconsistencies in the Executable Model that were not detected before. For this purpose, the information of the dynamic behavior of the model and its submodels over time needs to be accessible from outside, which usually requires some kind of instrumentation of the Executable Model's source code. Simply translating the raw model output data into a dynamic, visual animation, and pre-

senting it to experts familiar with the system and the problem, triggered the identification of errors or provided starting points for the search for additional errors. This combination of V&V techniques allows the human being to fully apply his mental abilities to detect "something wrong" in the model behavior. Thus, although face validation is a totally informal approach to V&V, it was found to be highly efficient, and will play as important a role in M&S V&V in the future as it did in the past.

6.5 Other Promising Paths Through the V&V Triangle

During evaluation of the V&V Triangle, only a small subset of the possible paths through the V&V Triangle was explored. Several others look promising and are briefly discussed in the following.

6.5.1 Instrumentation, Assertions, and Inductive Assertions

Instrumentation in the context of software testing means to insert statements into the code, which catch and record internal program runtime information that is otherwise hidden from the tester. The most efficient way to instrument a program depends on the test strategy and the aims of the test. However, a promising instrumentation strategy is to catch the values of parameters at the beginning and the end of a process block.
Assertions concerning the expected behavior of the model can be directly derived from the behavior properties identified in V&V sub-phases 1.1, 1.2, 2.1, and 2.2. In all V&V sub-phases 3.x, these behavior propositions can be used as assertions in the Formal Model, assuming that one has a modeling environment that detects assertions violations. In sub-phase 4.1 this instrumentation can be used to monitor the execution of any process block and detect, if a precondition stated as assertion is violated, and to ensure that a post-condition, again stated as an assertion, holds. If required, it can be proven that a precondition is strong enough to guarantee the post-condition after execution of the process block. In V&V sub-phase 4.3 assertions can be used to control the submodel state and to automatically report violations of the desired state.

6.5.2 Statistical Data Evaluation

Another approach to the V&V of a model or simulation results focuses exclusively on data evaluation. With the guarantee that a quantitative symbolic model works on quantitative input and produces quantitative output that can be compared to quantitative real data, truly objective statements about some model aspects can be made. In following the path of data evaluation, the model itself is considered to be a black box. If necessary, the values of the state variables are made visible as output parameters (instrumentation). Then all data used by, produced by, or related to the model is analyzed for particular statistical characteristics.
Methods provided by mathematical statistics enable data evaluation in the M&S domain. To handle large amounts of data (a *sample* in the following), mathematical statistics define characteristics as *frequency* (of a particular data value), *relative frequency*, and – for continuous data – *class frequency* and *relative class frequency*. The (relative) frequency of sample values can be graphically depicted in "*histogram*" form. The relative frequency allows the creation of an *empiric distribution function* and an *empiric cumulative distribution function.* For each data sample an *empiric mean*, *median*, *p-quantiles*, *standard deviation* and *variance* can be determined [Bosch 1987]. This allows one to categorize and compare large samples by their statistical characteristics, and supports the interpretation of the sample. [Sargent 1999] proposes the use of box plots to visualize the characteristics of data used in the context of M&S V&V.
The size of a sample plays an important role when characterizing it. On the one hand, if the sample size is too small, the characteristics are meaningless, on the other hand starting from a particular sample size, no more relevant information is gained by increasing the sample size.

More often than not there is only a small sample of real world data available (if any at all). The allowed *confidence* in the outcome of a *statistical test* or *hypothesis test* indicates the meaning of the comparison of data samples. Statistical methods can be used in the V&V sub-phases 3.2 (goodness of fit of distribution functions), 4.1 (correlation of random numbers, model data vs. real data comparison), and 5.1 (statistical significance of Simulation Results) to detect undesired correlations or to assess the similarity of data distributions in different samples.

6.5.3 Flow Analysis

Both the control and data flows of the Conceptual Model, the Formal Model, and the Executable Model show great promise for M&S V&V. Control flow analysis (and control flow testing) and data flow analysis (and data flow testing) can be copied from software testing.
According to [Beizer 1990] a control flow graph is a simplified graphical representation of the program's control structure. Its graphical elements are process blocks, decisions, and junctions. A process block is a sequence of program statements, uninterrupted by either decisions or junctions, with exactly one entry and one exit. A decision is a program point where the control flow diverges. A junction is a program point where the control flow merges.
The reasons for the creation of a program control flow graph are directly transferred to M&S V&V:

1. Control flow graphs visualize the control flows, and abstract from other program elements. This supports the identification of errors in the control flows.
2. The control flow graphs serve as a preparation for path testing.
3. Control flow graphs of the Conceptual Model, the Formal Model, and the Executable Model can be compared to conduct a consistency check.

In the V&V Triangle, starting with V&V sub-phase 2.1, a flow diagram can be extracted from the Conceptual Model and plotted in one or several control flow graphs, depicting all possible choices of "paths" within each submodel. This type of diagram focuses on decision making in the Conceptual Model. This graph can be reviewed during V&V sub-phase 2.2 forcing the reviewer to explicitly concentrate on the decision making description within the submodels. In V&V sub-phase 3.2 the control flows of the Conceptual Model and the Formal Model may be compared for consistency checking, which also can be done with the control flows of the Formal Model and the Executable Model in V&V sub-phase 4.2. Finally model test cases are created based on the Conceptual Model control flows for V&V sub-phase 4.3.
A data flow graph, which documents the modification of a data item during the program execution [Beizer 1990] can be used as a "object or submodel flow graph" in the V&V Triangle. In a nutshell, a data flow graph is a graph consisting of nodes and directed links (arrows). The nodes denote operations (assignment and mathematical operation), and the links denote that the result of the operation at the arrow's tail is needed for the operation at the arrow's head. This approach is transferable into the V&V Triangle: The object flow graphs or submodel flow graphs are extracted from the Conceptual Model in V&V sub-phase 2.1, depicting all manipulations of data elements (e.g., attributes, state variables). This type of diagram focuses on the initialization, modification, and recalculation of (internal) submodel values. This diagram may also be subjected to further analysis or be checked for violation of a specification, and is used during later V&V sub-phases (e.g., 4.3) to trace the life of a simulation object.

6.6 Concluding Remarks on the V&V Triangle

The V&V Triangle was derived from rigorous analysis of the V&V challenge itself, of existing approaches to V&V, of personal M&S experience, and was evaluated in a test case, and promises to be another little step toward the integration of V&V activities into model development. During refinement of the V&V Triangle specification, quickly the limits of generality

were reached. To create a V&V process independently from a particular application domain, independently from a modeling formalism, and independently from a technical realization environment, hardly enhances the detection of those V&V techniques, which are a real help to those confronted with a real V&V problem. Thus the author believes that any further refinement of the V&V Triangle without further reduction of the problem space is ineffective.
The key to V&V in everyday modeling practice is the availability of tools. Rigorous specification requirements, which are required for entering model documentation into a general model documentation framework force those involved in the modeling process into rethinking. However, the modeler needs to have a direct benefit from electronic model documentation, as, e.g., automatic generation of model documentation or automatic access to a model repository for submodels reuse. Then, with electronic documentation of the intermediate products available, automatic V&V activities such as consistency checking can be performed on these intermediate products. The V&V Triangle provides the foundation for the identification of the functional requirements for V&V support tools over the V&V subphases.
The V&V Triangle provides a detailed overview of the dependencies between the objectives and activities, which are important for the demonstration of the correctness and suitability of a model or simulation results. However, in such a wide research context, only a limited number of identified issues was explored in depth, with numerous open issues remaining. The most fascinating of them are outlined in the following.

6.7 V&V of Component-Based Models

To reduce the complexity of V&V of a model or simulation results, here the information available about the model is decomposed along the "integration/decomposition axis" (see Figure 9) of the three-dimensional model information space introduced in section 4.3.1. The explicit decomposability of a component-based model encourages the V&V of each component prior to the V&V of the overall composed model, which can then be reduced to the demonstration that the valid components interact in a valid manner. However, in the description of the V&V Triangle, model decomposability most often is only an implicit assumption, which is not explicitly discussed for most of its V&V sub-phases. With the growing popularity of the concepts of simulator interoperability (e.g., HLA federations, [Dahmann, Kuhl, and Weatherly 1998]) and component-based simulation [Hofmann 2003], the need for a methodology for the V&V of component-based models arises.
As a general approach to the V&V of component-based models, a bottom-up approach is recommended. Here the first step is the V&V of each component (submodel), followed by the layer-wise demonstration that their integration into the overall (composed) model also is correct and suitable. If a component is reused in another model, under certain (limited) circumstances also the associated V&V results can be reused: All verification results, which demonstrate the consistency among the intermediate products can be leveraged, as well as those verification results that demonstrate consistency with those precisely specified external rules that are still applicable. Validation results may also be reused, as long as the new intended purpose of model use is related closely enough to the original intended purpose.
The development of a methodology for V&V of component-based models requires an in-depth examination of

- the potential and limits of the integration of correct and suitable components into a higher level correct and suitable component,
- the impact of "black boxes" on the demonstration of overall model correctness and suitability, and
- the potential of reuse especially of validation results, if the intended purpose of the new model deviates from the original intended purpose of the component.

The V&V Triangle provides a stable foundation for this examination.

6.8 V&V of Distributed Simulations

An implicit assumption of the V&V Triangle is that any internal information of the Executable Model can be made available and monitored ("white box"), which in practice (especially with commercial models) often not is the case. If the Executable Model is not a monolithic implementation, but technically distributed, an additional opportunity for the instrumentation of the Executable Model and the monitoring of its internal state exists that is otherwise not available. The communication between the distributed submodels can be monitored (as long as it is unencrypted), which gives additional insight into the internal behavior of the overall model.

For the V&V of distributed simulations the research needs are similar to those for the V&V of component-based models. (Actually the distributed Executable Model is composed of executable submodels, although they do not need to be as flexible as one expects from a "model component".) Effective and efficient V&V techniques need to be identified or refined, which help to ensure that the distribution of the submodels of a correct and suitable overall model does not lead to its distortion or falsification. Also these techniques need to be embedded into an appropriate development process for distributed models, as, e.g., the FEDEP [IEEE 1516.3 2001].

6.9 Assessment of Credibility

V&V is conducted to demonstrate that a model or simulation results are correct and suitable, or, in other words, to increase their perceived correctness and suitability. Unfortunately, for non-trivial models under financial constraints it usually is not possible to actually *prove* that a model or simulation results are suitable and correct. As already discussed in section 4.4, to implement target-oriented V&V and to establish a sufficiently high degree of credibility of the model, the rigor and intensity of applicable V&V should be adapted accordingly. However, the problem of how to assess credibility still remains.

In section 4.4 credibility is discussed qualitatively, which was sufficient for the creation of the V&V Triangle. More objectivity would be added to the V&V process, if there was a *quantitative* measure of credibility available – initial ideas could be taken from Theory of Belief [Shafer 1976]. Alternatively, [Becker 1989] discusses quantitative models for software reliability, which also hold potential for transfer to M&S. However, the exact nature of a quantitative measure of credibility is difficult. A discussion about the quantification of credibility can be found in [Brade and Köster 2001; Brade, Maguire, and Lotz 2002].

The idea sketched in the following is based on the Claim-Argument-Evidence concept used in the ITOP on V&V (section 3.2.10) and the three-dimensional model information space (section 4.3.1). The credibility of the claim of the V&V agent that the model is suitable for the intended purpose will be judged by the *convincing force* of the logical decomposition of this general claim in situation-specific sub-claims (breadth), and the *probative force* of the evidence referred to for substantiation of the sub-claims (depth). In [Brade and Köster 2001] and [Brade, Maguire, and Lotz 2003] for the quantification of convincing force and probative force, linguistic variables (low, medium, and high) were chosen, but this definitely is not a final solution. Fuzzy logic could be used to put this approach on a more stable scientific foundation.

6.10 Tool Support

To make V&V affordable, tools are required, which support both model development and execution and V&V [Pace 1999c]. [Modeling and Simulation Information Analysis Center 2000] provides an overview of currently available tools, and numerous ideas for additional

V&V tools are born when reading this list. Proposals for tool development are found in the following.

6.10.1 Model Documentation System

As mentioned before, the availability of a model documentation system is desirable. There are numerous documentation management systems for storing electronic documents, but these systems do not support access to specific contents of the documentation. A model documentation system should be able to store the contents of an intermediate product according to its contents requirements given in the sections 2.2, 3.2, 4.2, 5.2, and 6.2 of Appendix A, respectively, and to access these contents individually. This typed data base then is a stable foundation for rigorous configuration control, checks for correctness in form, checks for internal consistency, and checks for IP-IP traceability (especially after the modification of any intermediate product). This model documentation system should be coupled with the tools used for the creation of the intermediate product via an open standard like XML. A first step into this direction is documented in [Moritz 2002].

6.10.2 An Integrated Model Development System

As long as there is no direct benefit in changing a specification prior to the modification of the source code, for minor changes software developers first modify their code before they adapt the design documents (if they are adapted at all). After a while, minor changes become major changes, and code and design become inconsistent. This situation changed with the usage of integrated development systems as [Borland 2003], where changes in the UML design directly impact the code, and vice versa.
The potential of something similar for model development should be explored. A flexible modeling formalism like DEVS [Zeigler, Praehofer, and Kim 2000] could be used as a bridge between the Conceptual Model, which documents the abstraction and idealization performed, and the Executable Model, which is finally used for the generation of Simulation Results. A first step in this direction is made with a visual modeling environment for DEVS [Schäfer 2003].

6.10.3 Output Analyzers

Models usually produce vast amounts of output, which somehow need to be analyzed. Tools for statistical analysis exist, but the use of mathematical logics for the detection of undesired entries in the I/S/O trace can be increased. With VIOLADE [Brade and Waldner 2003] a tool for the automatic detection of desired behavior violations was created. There still remain numerous interesting tasks, including the systematic identification of typical behavior problems in monolithic models and federations, the identification and eventually modification of sufficiently expressive mathematical logics for behavior specification, a methodology for the creation of acceptance automata for these logics, and the implementation of the prototypical demonstrator.

REFERENCES

Arthur, J.D. and R.E. Nance. 2000. V&V Without Independence: A Recipe for Failure. In Proceedings of the 2000 Winter Simulation Conference, Society for Modeling and Computer-based simulation International.

Balci, O. 1990. Guidelines for Successful Simulation Studies, Proceedings of the 1990 Winter Simulation Conference, Society for Modeling and Computer-based simulation International.

Balci, O. 1997a. Principles of Simulation Model Validation, Verification, and Testing. Transactions of the SCS International, Volume 14, Number 1, pp. 03-12. ISSN 0740-6797/97.

Balci, O. 1997b. Verification, Validation and Accreditation of Simulation Models. Proceedings of the 1997 Winter Simulation Conference, Society for Modeling and Computer-based simulation International.

Balci, O. 1998a. Verification, Validation and Testing. In: The Handbook of Simulation edited by J. Banks, Wiley, New York.

Balci, O. 1998b. Verification, Validation, and Accreditation. In Proceedings of the 1998 Winter Simulation Conference, pp. 41-48, Society for Modeling and Computer-based simulation International.

Balci, O. 2000. Well-Defined Intended Uses: An Explicit Requirement for Accreditation of Modeling and Simulation Applications, Proceedings of the 2000 Winter Simulation Conference, pp. 849-854, Society for Modeling and Computer-based simulation International.

Balci, O. 2001. A Methodology for Certification of Modeling and Simulation Applications. In ACM Transactions on Modeling and Computer-based simulation 11 (4), pp 352-377.

Banks, J. (ed.). 1998. Handbook of Simulation, Wiley, New York.

Becker, G. 1989. Softwarezuverlässigkeit – Quantitative Modelle und Nachweisverfahren. Walter de Gruyter Verlag Berlin, New York. ISBN: 3110122278.

Beizer, B. 1990. Software Testing Techniques (Second Edition). Van Nostrand Reinhold, New York. ISBN: 0-442-20672-0.

Berghammer, R. 2001. Semantik von Programmiersprachen. Logos Verlag Berlin. ISBN: 3-89722-489-5

Bienmüller, T., W. Damm, J. Klose, and H. Wittke. 2001. Formale Analyse und Verifikation von Statemate-Entwürfen. In it+ti 43 1/2001, Oldenbourg Verlag, pp. 29 – 34, ISSN 0944-2774.

Boehm, B.W. 1981. Software Engineering Economics. Prentice Hall, Inc., Eaglewood Cliffs, New Jersey. ISBN 0-13-822122-7

Borland. 2003. Together. http://www.borland.com/together/ .

Bosch, K. Elementare Einführung in die angewandte Statistik. Vieweg, Braunschweig.1987

Brade, D. 2000. Enhancing Modeling and Simulation Accreditation by Structuring Verification and Validation Results, Proceedings of the 2000 Winter Simulation Conference, pp. 840-848, Society for Modeling and Computer-based simulation International.

Brade, D. 2002. Formal Verification of HLA Federations Using Temporal Logic. In: Pro-

ceedings of the 16th European Simulation Multiconference, Darmstadt, pp 273 - 277, Society for Modeling and Computer-based simulation International.
Brade, D. and A. Köster. 2001. Risk-based Validation & Verification Levels Definition. Proceedings of the European Simulation Interoperability Workshop, London, Simulation Interoperability Standardization Organization.
Brade, D. and A. Lehmann. 2002. Model Validation and Verification. In Modeling and Simulation Environment for Satellite and Terrestrial Communication Networks – Proceedings of the European COST Telecommunication Symposium, pp.281 - 299. Kluwer Academic Publishers of Boston, USA. ISBN 0-7923-7547-5
Brade, D. and C. Waldner. 2003. Automatic Detection of Behavior Specification Violations. In Proceedings of the 03 European Simulation Interoperability Workshop, Stockholm, organized by the Simulation Interoperability Standardization Organization.
Brade, D. and I. Cegla. 2003. Conceptual Modeling Meets Formalization. In Proceedings of the 03 Spring Simulation Interoperability Workshop, Orlando, organized by the Simulation Interoperability Standardization Organization.
Brade, D., R. Maguire, and H.-B. Lotz. 2002. Arguments-based Credibility Levels. In Proceedings of the SISO Spring Simulation Interoperability Workshop, Orlando, Simulation Interoperability Standardization Organization.
Brain, C. 2001. Achieving the Right Level of Credibility, Presentation at the JASA Workshop: Assuring M&S Credibility for Defense Acquisition and T&E, Reno.
Brecht, S. and K. Riepl. 2002. Performability Analyse von Internet-Routing-Strategien. Studienarbeit IS 15/2002, Fakultät für Informatik, Universität der Bundeswehr München, Neubiberg.
Brendel R. und A. Schäfer. 2002. Performability-Analyse von Routing-Strategien mittels analytisch-numerischer Verfahren. Studienarbeit IS 16/2002 an der Fakultät für Informatik, Universität der Bundeswehr München, Neubiberg.
Bundesamt für Wehrtechnik und Beschaffung. 2000. Verbesserung der Bedarfsdeckung durch den Einsatz von Simulation. Entwurf Teilkonzept Simulation Rüstung. Bundesamt für Wehrtechnik und Beschaffung.
Bundesministerium für Verteidigung. 1997. Allgemeiner Umdruck 250 „Entwicklungsstandard für IT-Systeme des Bundes – Vorgehensmodell“.
Bundesministerium für Verteidigung. 2000. Leitlinie für Modellbildung und Simulation in der Bundeswehr. Version 1.0, Bundesministerium für Verteidigung.
Burch, J.R., E.M. Clarke, D.E. Long, K.L. McMillan, and D.L. Dill. 1994. Symbolic Model Checking for Sequential Circuit Verification, IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 13, No. 4, pp. 401–424.
Carr, T.J. III and O. Balci. 2000. Verification and Validation of Object-Oriented Artifacts Throughout the Simulation Model Development Life Cycle. In Proceedings of the SCS Winter Simulation Conference, Orlando, Society for Modeling and Computer-based simulation International.
Conwell, C.L., R. Enright, and M.A. Stutzman. 2000. Capability Maturity Models Support of Modeling and Simulation Verification, Validation, and Accreditation. In Proceedings of the 2000 Winter Simulation Conference, Orlando, Society for Computer Simulation International.
Courtois, P. J. 1981. Decomposability. Academic Press.
Dahmann, J.S., F. Kuhl, and R. Weatherly. 1998. Standards for Simulation: As Simple As Possible But Not Simpler. The High Level Architecture For Simulation. SIMULATION 71:6,7 p. 378-387.
Davis, P.K. 1992. Generalizing Concepts and Methods of Verification, Validation, and Accreditation (VV&A) for Military Simulations. Report published by RAND, Santa Monica, CA. ISBN 0-8330-1298-3.

Defense Modeling and Simulation Office. 1996. Department of Defense Verification, Validation and Accreditation (VV&A) Recommended Practices Guide, Defense Modeling and Simulation Office, Alexandria, VA. Co-authored by: O. Balci, P. A. Glasgow, P. Muessing, E. H. Page, J. Sikora, S. Solick, and S. Youngblood. USA.

Defense Modeling and Simulation Office. 1999a. High Level Architecture – Federation Development and Execution Process (FEDEP) Model. Version 1.5.

Defense Modeling and Simulation Office. 1999b. High Level Architecture – Federation Development and Execution Process (FEDEP) Checklists. Version 1.5.

Defense Modeling and Simulation Office. 2000. Verification, Validation and Accreditation (VV&A) Recommended Practices Guide (RPG). Alexandria, VA. http://vva.dmso.mil/ .

Defense Modeling and Simulation Office. 2003. Functional Description of the Mission Space. USA. https://www.dmso.mil/public/transition/fdms/ .

DeMillo, R.A., W.M. McCracken, R.J. Martin, and J.F Passafiume. 1987. Software Testing and Evaluation. The Benjamin/Cummings Publishing Company, Inc. ISBN: 0-8053-2535-2.

Department of Defense. 1995. Modeling and Simulation Master Plan. USA.

Department of Defense. 1996. Instruction 5000.61: US Department of Defense Modeling and Simulations Verification, Validation, and Accreditation. USA.

Department of the Army, Corps of Engineers. 1957. The San Francisco Bay Model. USA. http://www.spn.usace.army.mil/bmvc/ .

Department of the Army, Headquarters. 1999a. Test and Evaluation: M&S VV&A Methodology, TECOM Pamphlet 73-4, U. S. Army Test and Evaluation Command, Aberdeen Proving Ground, Maryland.

Department of the Army, Headquarters. 1999b. Pamphlet 5-11: VV&A of Army Models and Simulations. Washington, DC.

DEVSJAVA 2.7. 2003. DEVSJAVA. Arizona Center for Integrative Modeling and Simulation. http://www.acims.arizona.edu/software/software.shtml .

Dompke, U. 1999. Report on NATO Long Term Scientific Study on Human Behavior Representation. Proceedings of the 8th CGF Conference 1999,

Echtle, K. 1990. Fehlertoleranzverfahren. Springer Verlag Berlin. 3-540-52680-3

Edelman, E. 1997. The Mathematics of the Pentium Bug. In SIAM Rev., Vol. 39, No. 1, pp. 54 – 67. Society for Industrial and Applied Mathematics.

Epstein, J.M. 1997. The role of the global positioning system in the environment. New York University Environmental Law Journal, Vol. 6, Issue 1, pp 72. – 92.

Flight Mechanics Panel Working Group WG-12 on Validation of Missile Simulation. 1985. Final Report, AGARD Advisory Report No. 206.

Fowler, M. 1997. UML Distilled – Applying the Standard Object Modeling Language. Addison-Wesley.

Frankfurter Allgemeine Zeitung. 1995. Amerikanische Behörde warnt vor Fehlern in der Naviagationselektronik, in Frankfurter Allgemeine Zeitung, 12. August.

Fujimoto, R.M. 2000. Parallel and Distributed Simulation Systems. John Wiley & Sons, Inc.

Gabor, Dietz, and Grahn. 2000. Konzept zur Validierung, Verifizierung und Akkreditierung von Gefechtssimulationssystemen am Beispiel SiRa Brig. Amt für Studien und Übungen der Bundeswehr.

Geilen, M. 2000. Model-Checking in Simulations of Distributed Systems. In Proceedings of the 12th European Simulation Symposium, pp. 606 – 611.

General Accounting Office. 1994. New Denver Airport: Impact of the delayed baggage system. Briefing report, RCED-95-35BR. USA.

Gerth R., D. Peled, M. Vardi, and P. Wolper. 1995. Simple On-the-fly Automatic Verification of Linear Temporal Logic. In Proceedings of the 13th Symposium on Protocol Specification, Testing and Verification, pages 3-18.

Glinsky, E. and G. Wainer. 2002. Definition of Real-Time Simulation in the CD++ Toolkit. In Proceedings of the 2002 Summer Computer Simulation Conference, San Diego, Society for Modeling and Computer-based simulation International.

Graffagnini, J., S. Youngblood, and R. Lewis. 1999. An Overview of the Verification, Validation, and Accreditation (VV&A) Process for the HLA FEDEP. In the Proceedings of the SCS 1999 Winter Simulation Conference, pp 421 – 428, Society for Modeling and Computer-based simulation International.

Gutte, Schmidt, Schäfert, and Schmatz. 2001. Studienabschlußbericht: Dokumentation von Simulationsmodellen SKZ 12990 Y 016 W. Amt für Studien und Übungen der Bundeswehr.

Harmon, S.Y., D.C. Gross, and S.M. Youngblood. 1999. Why Validation? In Proceedings of the 1999 Summer Computer Simulation Conference, Society for Computer Simulation International.

Hellekalek, P. 1997. Good Random Number Generators are (not so) Easy to Find. In: I. Troch, F. Breitenecker, Eds., Proceedings of the 2nd Int. Symposium on Mathematical Modelling (2nd MATHMOD VIENNA), pp. 1-15, ARGESIM Report No. 11, Technische Universität Wien, Österreich.

Henriksen, J.O. 1998. Stretching the Boundaries of Simulation Software. In Proceedings of the 1998 Winter Simulation Conference, Orlando, Society for Computer Simulation International.

Hetzel, W. 1984. The Complete Guide to Software Testing. QED Information Sciences, Wellesly Mass

Hiller, F.S. and G.J. Liebermann. 1997. Operations Research – Einführung. 5. Auflage, Oldenbourg-Verlag München Wien, ISBN: 3-486-23987-2.

Hofmann, M. 2003. On Decomposition: Some Fundamental Aspects in Model Development. In Proceedings of the SISO European Simulation Interoperability Workshop, Simulation Interoperability Standardization Organization.

Hohl, S. 2001. Graphische Modellierung von HLA-Föderationen. Diplomarbeit ID 37/2001, Fakultät für Informatik, Universität der Bundeswehr München, Neubiberg.

Hong, G.P. and T.G. Kim. 1996. A Framework for Verifying Discrete Event Models Within a DEVS-Based System Development Methodology. In Transactions of The Society for Computer Simulation, Volume 13, No. 1, pp. 19-34. ISSN 0740-6797/96.

IEEE 1516. 2001. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA). IEEE 1516.1 through 1516.3. Institute of Electrical and Electronical Engineers.

IEEE 610.3. 1989. IEEE Standard for Modeling and Simulation (M&S) Terminology. IEEE 610.3. Institute of Electrical and Electronical Engineers.

International Test Operations Procedure on V&V Working Group of Experts. 2003 (to appear). International Test Operations Procedure on V&V. Draft, ITSEC, MC7, WG7.2.

Katoen, J.P. 1999. Concepts, Algorithms and Tools for Model Checking. Arbeitsberichte des Instituts für mathematische Maschinen und Datenverarbeitung (Informatik), Friedrich-Alexander-Universität Erlangen-Nürnberg, Band 32, Nummer 1.

Keppler, K. 1993. Random Variables Made Simply. In: Computer Language, pp. 67-79.

Kilikauskas, M. 2001. Obstacles to Success in VV&A Efforts and How to Overcome Them. In Proceedings of the 15th European Simulation Multi-Conference 2001, Prague, edited by Kerkhoffs and Snorek, pp. 134 – 138. ISBN: 1-56555-225-3.

Kilikauskas, M.L., D. Brade, R.M. Gravitz, D.H. Hall, M.L. Hoppus, R.L. Ketcham, R.O. Lewis, and M.L. Metz. 2002. Estimating V&V Resource Requirements And Schedule Impact, in Dale K. Pace (editor), V&V State of the Art: Proceedings of Foundations '02, a Workshop on Model and Simulation Verification and Validation for the 21st Century, October 22-24, 2002, Laurel, MD. CD published by The Society for Modeling and Simula-

tion.
Kim, J.-H., H.S. Park, and D.-K. Baik. 2000. Formal Modeling and Verification of an Information Retrieval System using SMV.
Kleijnen, J.P.C. 1998. Experiment design for Sensivity Analysis, Optimization, and Validation of Simulation Models. In: The Handbook of Simulation edited by J. Banks, Wiley, New York.
Kleijnen, J.P.C. 1999. Validation of Models: Statistical Techniques and Data Availability. In Proceedings of the 1999 Winter Simulation Conference, pp. 647-654, Society for Computer Simulation International.
Klir, G. 1985. Architecture of Systems Complexity. Sauders, New York.
Könke, D. et. al. 2000. Dokumentation Simulationsmodelle. Abschlussbericht zur gleichnamigen Studie, ITIS, Neubiberg.
Kruchten, P. 2000. The Rational Unified Process: An Introduction (2nd Edition). Addison-Wesley Pub Co. ISBN: 0201707101.
Kuhl, F., R. Weatherly, and J. Dahmann. 2000. Creating Computer-based simulation Systems – An Introduction to the High Level Architecture. Prentice Hall.
L'Ecuyer, P. 1998. Random Number Generation. In: The Handbook of Simulation, edited by Jerry Banks, John Wiley and Sons.
Lamprecht, G. 1976. Einführung in die Programmiersprache SIMULA. Vieweg, Braunschweig. ISBN 3-528-03321-5
Lee, S. and R.M. O'Keefe. 1994. Developing a Strategy for Expert System Verification and Validation. In IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 4, pp. 643 – 655.
Lehmann, A., C. Berchtold, D. Brade, M. Hofmann, and T. Krieger. 2002. Verifizierung, Validierung und Akkreditierung von Modellen und Simulationen, technischer Anteil. Final Report, Study: E/F11B/Z0217/T5228. Institut für Technik Intelligenter Systeme an der Universität der Bundeswehr München.
Lemessi, M. 2001. An SLX-Based Microsimulation Model for a two-lane Road Section. In Proceedings of the 2001 Winter Simulation Conference, Society for Computer Simulation International.
Liggesmeyer, P., H.M. Sneed, and A. Spillner (Hrsg.). 1992. Testen, Analysieren und Verifizieren von Software. Informatik Aktuell, Springer Verlag.
Lindemann, C. 1998. Performance Modeling with Deterministic and Stochastic Petri Nets. John Wiley and Sons, New York. ISBN 0-471-97646-6.
Merriam Webster Online. 2003. The Language Center. http://www.m-w.com/home.htm .
Modeling and Simulation Information Analysis Center (MSIAC). 2000. Verification, Validation, and Accreditation (VV&A) Automated Support Tools – A State of the Art Report.
Morales, M.C. and M.R. Moulding. 2000. Improving the Verification of Synthetic Environments: Advanced Checking of HLA Specifications across a Federation. In Proceedings of the SISO Spring Simulation Interoperability Workshop, Simulation Interoperability Standardization Organization.
Moritz, S. 2003. Intelligente Führung durch den Prozess der Verifikation und Validierung von Simulationsmodellen. Diplomarbeit 29/2002, Fakultät für Informatik, Universität der Bundeswehr München, Neubiberg.
Muessing, P., D. R. Laack, and J. J. Wrobleski, Jr. 1997. Optimizing the Selection of VV&A Activities: A Risk/Benefit Approach, Proceedings of the 1997 Summer Computer Simulation Conference, Society for Computer Simulation International.
Mugridge, C. 1999. Verification, Validation and Accreditation of Models and Simulations Used for Test and Evaluation – a Risk/Benefit Based Approach. Internal Report, Technical Development Group, Defense Evaluation and Research Agency UK.
Myers, G.J. 1979. The Art of Software Testing. John Wiley and Sons.

North Atlantic Treaty Organization. 1998. NATO Modelling and Simulation Master Plan. Document AC/323 (SGMS)D/2, Version 1.0.

Object Management Group. 1999. Unified Modeling Language (UML) V1.3 Alpha. OMG Headquaters, Framingham.

Office of Safety and Mission Assurance. 1993. Software Formal Inspections Guidebook. National Aeronautics and Space Administration. NASA-GB-A302, Washington DC.

Pace, D.K. 1999a. Conceptual Model Descriptions, In Proceedings of the 1999 Summer Computer Simulation Conference, pp. 429-434, Society for Computer Simulation International.

Pace, D.K. 1999b. SIMVAL99. In Proceedings of the 1999 Summer Computer Simulation Conference, pp. 405-408, Society for Computer Simulation International.

Pace, D.K. 1999c. V&V Technology. Proceedings of the 1999 Summer Computer Simulation Conference, pp. 441-445, Society for Computer Simulation International.

Pace, D.K. 2003. The Roles of Metrics in Simulation Verification and Validation. In Proceedings of the SISO Spring Simulation Interoperability Workshop, Orlando, Simulation Interoperability Standardization Organization.

Partsch, H. 1991. Requirements Engineering. Oldenbourg Verlag. ISBN: 3-486-20784-9.

Paulk, M.C., B. Curtis, M.B. Chrissis, and C.V. Weber. 1993. Technical Report: „Capability Maturity Model for Software, Version 1.1“ CMU/SEI-93-TR-024, ESC-TR-93-177. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Pawlikowski, K., H.-D.J. Jeong, and J.-S. Lee. 2002. On Credibility of Simulation Studies of Telecommunication Networks. In IEEE Communications Magazine.

Peterson, I. 1997. Pentium Bug Revisited. In MAA online. The Mathematical Association of America. http://www.maa.org/mathland/mathland_5_12.html

Popper, K.R. 1993. Objektive Erkenntnis: Ein evolutionärer Entwurf. Hamburg (Objective Knowledge).

Posner, M.I., 1988. Introduction: What is it to be an expert? In M.T.H. Chi, R. Glaser, M.J. Farr (Eds.): The nature of Expertise. Hilldale NJ: Erlbaum; xxix – xxxvi

Pygott, C. and S. Wilson. 1996. Uses of an Argument Framework; its Data Model and Generic Features. Technical Report DRA/CIS3/PROJ/A53XL/95011/1.0, Defence Research Agency, Farnborough, Hampshire GU14 6TD, UK.

Robinson, S. 1999. Simulation Verification, Validation, and Confidence: A Tutorial. In Transactions of the Society for Computer Simulation International, Volume 16, Number 2, pp 63-69. ISSN 0740-6797/99

Rosenberg, L.H. 2001. Technical Report: Verification and Validation Implementation at NASA. NASA, Goddard Space Flight Center.

Ruf, J. and T. Kropf. 2001. Formale Verifikation diskreter Echtzeitsysteme. In it+ti 43 1/2001, Oldenbourg Verlag, pp. 39 – 46, ISSN 0944-2774.

Sahner, R.A., K.S. Trivedi, and A. Puliafito. 1998. Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package. Kluwer Academic Publishers.

Sargent, R.G. 1999. Validation and Verification of Simulation Models. In Proceedings of the 1999 Winter Simulation Conference, pp. 39-48, Society for Computer Simulation International.

Schäfer, A. 2003. Visuelle Darstellung von DEVS-Modellen. Diplomarbeit, Institut für Technische Informatik, Universität der Bundeswehr München, Neubiberg.

Schmidt, B. 1985. Fachberichte Simulation: Systemanalyse und Modellbau – Grundlagen der Simulationstechnik. Springer-Verlag.

Schnepf, L. 1990. Validation von Simulationsmodellen zur Leistungsanalyse von Rechensystemen: Motivation, Ansätze, Aufgabenstellung. Interner Bericht, Neubiberg, UniBw München, Fakultät für Informatik.

Scholten, A. 1998. Dokumentation von OR-Modellen: Leitfaden zur Erstellung einer Dokumentation. Amt für Studien und Übungen der Bundeswehr, Bereich Operations Research.

Schultheiß, V. 2001. Konzeption und Implementierung eines Werkzeugs zur Unterstützung der Verifikation, Validierung, und Akkreditierung von Simulationsmodellen. Studienarbeit 23/2001, Fakultät für Informatik, Universität der Bundeswehr München, Neubiberg.

Scrudder, R., W. Waite, M. Richardson, and R. Lutz. 1998. Graphical Presentation of the Federation Development and Execution Process. Technical Report, Defense Modeling and Simulation Office, USA.

Shafer G. 1976. A Mathematical Theory of Evidence, Princeton University Press.

Shannon, R.E. 1975. Systems Simulation and the Art of Science. Prentice Hall, Eaglewood Cliffs, N.J.

Simon, H. 1962. The Architecture of Complexity, Proceedings of the American Philosophical Society, Philadelphia, Vol 106, pp. 467-482.

Smith, D.J. and K.B. Wood. 1987. Engineering Quality Software. A Review of Current Practices, Standards, and Guidelines Including New Methods and Development Tools. Elsevier Applied Science.

SMV 2.5. 1998. The Symbolic Model Verifier. Model Checking Group, Carnigie Mellon University. http://www-2.cs.cmu.edu/~modelcheck/ .

Thaller, G.E. 2000. ISO9001 – Software-Entwicklung in der Praxis, 2. Auflage. Verlag Heinz Heise GmbH und Co. KG, Hannover.

Tittelbach, S. 2001. Die Staukiller. In it_science 1_2001.

Topp, H.H et. al. 1994. Haltestellenformen an innerörtlichen Hauptverkehrsstraßen, Berichte der Bundesanstalt für Straßenwesen, Verkehrstechnik, Heft V 12, Bergisch Gladbach.

Vangheluwe, H. 2001. Multi-Formalism Modelling and Simulation. PhD Thesis, University Gent.

Vincent, S. 1998. Input Data Analysis. In: The Handbook of Simulation, edited by Jerry Banks, John Wiley and Sons.

Wolverine Software. 2003. http://www.wolverinesoftware.com/wolverin.htm .

Woodcock, J. and J. Davies. 1996. Using Z, Specification, Refinement, and Proof. Prentice-Hall.

Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. Theory of Modeling and Simulation. Second Edition. Academic Press. ISBN: 0-12-778455-1

INDEX OF FIGURES AND TABLES

Figure 1: Embedded representation forms of a model 6
Figure 2: The chosen model development and execution process 11
Figure 3: Dependencies between V&V related terms 14
Figure 4: Sketch of the SW-CMM levels [Paulk et. al. 1993] 28
Figure 5: The CLIMB and their comparison data 33
Figure 6: Indicators hierarchy from [Balci 1990] 37
Figure 7: An abstract risk classification process 45
Figure 8: The three-dimensional model information space 48
Figure 9: Activities along the axes of the three-dimensional model information space 49
Figure 10: The model development and execution process of section 1.4.6 alternatively represented 51
Figure 11: Credibility of simulation results 52
Figure 12: Abstract representation of the credibility building process, based on the V&V results documentation (evidence) 53
Figure 13: V&V process framework 60
Figure 14: The V&V Triangle 62
Figure 15: Procedure for planning based on the V&V Triangle. 69
Figure 16: A tailored V&V process in V-form 70
Figure 17: Path of correctness in form and internal consistency 76
Figure 18: Path of behavior propositions violation detection 85
Figure 19: Acceptance automaton $A_{stop\_at\_lights}$ 88
Figure 20: The path of human review 93

Table 1: Impact domains and severity categories from [Mugridge 1997] 35
Table 2: Levels of system knowledge 45

APPENDICES

Appendix A: The V&V Triangle Specification

This appendix contains the detailed specification of the V&V Triangle. An overview over its ideas and concepts is provided in the main document, section 4.7. All references and acronyms are resolved in the main document.

TABLE OF CONTENTS

1 PREREQUISITE: SPONSOR NEEDS 1

2 V&V OF THE STRUCTURED PROBLEM DESCRIPTION 2

2.1 Expected Problem Definition Activities 3

2.2 Documentation Requirements for the SPD 4

2.3 Potential Errors in the SPD 5

2.4 SPD Internal V&V 6

2.5 SN to SPD Transformation V&V 8

3 V&V OF THE CONCEPTUAL MODEL 10

3.1 Expected System Analysis Activities 10

3.2 CM Documentation Requirements 12

3.3 Potential Errors in the CM 13

3.4 CM Internal V&V 15

3.5 SPD to CM transformation V&V 18

3.6 SN to CM transformation V&V 20

4 V&V OF THE FORMAL MODEL 22

4.1 Expected Formalization Activities 22

4.2 FM Documentation Requirements 24

4.3 Potential Errors in the FM 25

4.4 FM Internal V&V 26

4.5 CM to FM Transformation V&V 28

4.6 SPD to FM Transformation V&V 30

4.7 SN to FM Transformation V&V 32

5 V&V OF THE EXECUTABLE MODEL 34

5.1 Expected Implementation Activities 34

5.2 EM Documentation Requirements 36

5.3 Potential Errors in the EM 37

5.4 EM Internal V&V 38

5.5 FM to EM Transformation V&V 41

5.6 CM to EM Transformation V&V 43

5.7 SPD to EM Transformation V&V 46

5.8 SN to EM Transformation V&V 48

6 V&V OF SIMULATION RESULTS 50

6.1 Expected Experimentation Activities 50

6.2 SR Documentation Requirements 51

6.3 Potential Errors in the SR 52

6.4 SR Internal V&V 52

6.5 EM to SR Transformation V&V 55

6.6 FM to SR Transformation V&V 56

6.7 CM to SR Transformation V&V 58

6.8 SPD to SR Transformation V&V 60

6.9 SN to SR Transformation V&V 61

7 INDEX OF FIGURES 63

1 PREREQUISITE: SPONSOR NEEDS

The motivation for the development of a model originates from a specific task within an application domain. For various reasons introduced in the main document (sections 1.2 and 1.3.2), the examination or use of a real system needs to be supported or replaced by a quantitative modeling approach. The model or simulation results are intended to provide (additional) aid to supply some higher level need, and this need must be communicated. Such needs include reducing the education time for military pilots (training), or exploring the safety features of a public building (analysis). Often the need is the duplication of any type of real experiment. Thus, whenever the term "experimentation" is used in the following, other needs (such as training) are not explicitly excluded.

People in the application domain of the real system ideally are experts in their field, but one cannot expect them also to be M&S experts. The sponsor may have clear and precise ideas how to use the model or simulations results, but may not be able to communicate them to the developer. This is delicious, as finally the suitability of the model or simulation results is assessed with respect to the SN and the intended purpose of modeling or simulation. Therefore, already for the documentation of the SN here guidance is provided.

The SN express what the sponsor *expects to get*. They are provided by the sponsor/beneficiary, address mainly the system analyst and the modeler, and serve as essential input to the problem definition phase. Their representation is formless with respect to the model development process, but structured documents are recommended. To support the sponsor in providing all relevant input information to the model development process, a template for the documentation of the needs may be helpful. To be a useful product for the initiation of V&V in the model development process, the SN should contain the following entries (a sample "Initial Product: SN" is found in Appendix C):

1) **Configuration control information**: Required for configuration control and reference by succeeding IP, includes an unique identifier.
2) **Purpose and background**: Reasons for developing the model, including:
   a) **General problem set**: Description of the general problem that is intended to be solved by the use of M&S.
   b) **Experimentation aim**: Description of the purpose of conducting real experiments, if experimentation with or modification of the real system was possible.
   c) **Motivation for M&S**: Documentation of the reasons why M&S are expected to substitute or supplement the real experiments successfully.
   d) **M&S aim and intended purpose**: Description to which extend/coverage the aim of the real experiments shall be covered by M&S.
3) **Experimental framework description**: Detailed description of the real experiment or real process that simulation shall replace, including:
   a) **Idealized system and sample experiment description**: Description of the a priori known structure and behavior of the real system and its subsystems, desired instru-

mentation (locations and functionality of observation or measuring devices), and desired observation and modification of external influences. Identification of requirements that would go along with the real experiment and qualitative description of the results that are expected.

   b) **Influence parameters**: The identification of metrics for recording external influences on the system behavior, as used for the real experiment.
   c) **Goal parameters**: The identification of metrics of the real experiment for recording system behavior.
   d) **Acceptable limitations**: A priori identified irrelevant influences, rough estimate of the allowed differences between the model and the real system.

4) **Operation environment**: Description of the operational environment in, which the model finally will be used to generate SR (interaction with other systems or human), including:
   a) **Operation platform, peripheral devices, and operating system**: Description of the technical environment in which the simulation will be run.
   b) **Interoperability (Machine-Machine-Interfaces)**: Required interoperability with other hardware subsystems or EMs, including communication protocols and interfaces.
   c) **Interactivity (Man-Machine-Interfaces)**: The required possibilities of the manipulation of the simulation experiment by human operators (interactivity) must be considered in the model.
5) **Worst case impacts statement**: Identification of the worst case consequences of wrong decisions that may be derived from an unsuitable or incorrect model or simulation results.

The identification of all of the needs mentioned above is essential for an aim-oriented approach to model development and simulation experiment execution. The less of them are explicitly stated in the beginning, the less straight forward model development will be. If the sponsor is not able to provide the above information, degrees of freedom are introduced into model development, which are most likely to be undesired. Before starting to precisely define the requirements for modeling, the above information items should be identified, which may require a kind of “pre-phase” with support of a modeling expert. To ensure that the SN are internally consistent and actually reflect the problem of interest is the sponsor’s responsibility and beyond the tasks of an M&S V&V process.

2 V&V OF THE STRUCTURED PROBLEM DESCRIPTION

To allow meaningful verification and validation during the early stages of model development specific, accurate and detailed requirements that the model *must* meet should be derived from the SN, including the intended purpose and risk analysis. The SPD contains the precise and unambiguous requirements specification for further model development, including conceptual, formal, technical and experimental requirements as far as already identified. It lays the foundation for model development. The more vague the definition of the intended purpose of modeling or simulation, the less can later be said about the fitness for purpose of the model or the SR.

2.1 Expected Problem Definition Activities

The SPD is created during the problem definition phase, when the SN are analyzed and discussed with the aim to create a precise specification for model development. Here no guidance is provided on how to actually perform the problem definition, as it is strongly task dependent (see also main document, section 1.4.6). How to proceed for the creation of an SPD is left to the developer who may refer to [Shannon 1975] or [Partsch 1991]. Only these activities, which directly impact V&V are briefly outlined.

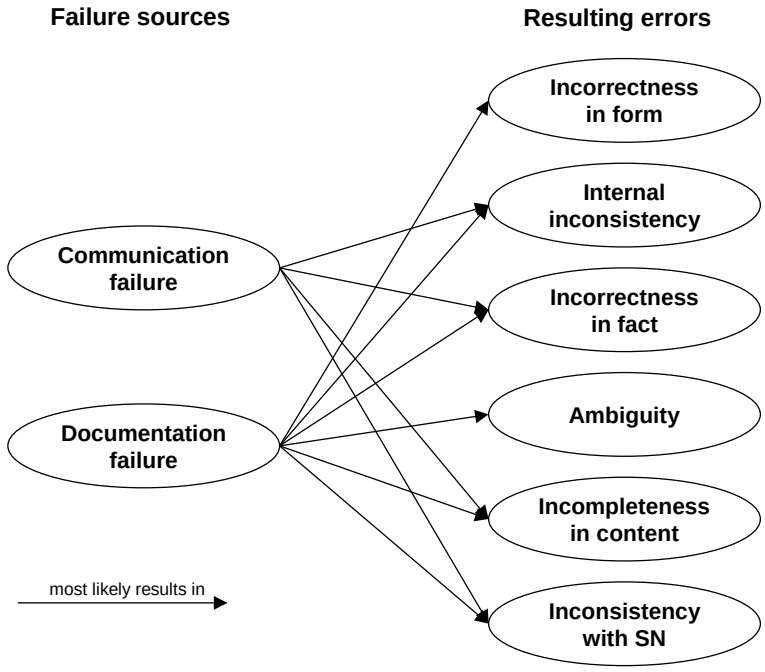


Figure 1: Dependencies between failures in the problem definition phase and errors in the SPD

One of the most important steps of the problem definition phase is to create an "M&S objectives hierarchy". As the intended purpose of the model or simulation results defined by the sponsor might be too complex or abstract, it should be decomposed into several clearly identified intended sub-purposes or M&S objectives. Starting with the overall SN, these should be decomposed into sub-needs, until precise requirements for M&S (M&S objectives), including test and acceptability criteria, can be derived from each sub-need. If possible, these requirements should be prioritized.

Errors in the SPD are likely to lead to the development of a model that is not suitable for the solution of the actual problem. ([Balci 1990] stresses the significance of this problem by the introduction of a "type III error".) They are caused by insufficiently pertinent questioning when identifying the intended purpose and required features of the model, or simply by communication problems. The following failures during problem definition are identified:

- *Communication Failure*: The problem definition should take place in the form of a dialogue between user, system analyst, and modeler. Both system analyst and modeler interpret and refine the SN for the purpose of modeling giving consideration to the intended purpose. The user may fail to communicate the needs.
- *Documentation Failure*: The SPD is written by the modeling expert, in close cooperation with the system analyst and the user, and documents *what the user will get*. If the partners fail to completely document the requirements, this might cause serious problems afterwards.

The problem definition phase is re-entered, when the SN change or an error in the SPD is detected.

2.2 Documentation Requirements for the SPD

Literature analysis [Shannon 1975; Banks 1998; Zeigler, Praehofer, and Kim 2000; Partsch 1991], numerous discussions with M&S experts, and own experience with simulation projects (including [Hohl 2001; Brecht and Riepl 2002] and Appendix C) yielded the need for V&V to record particular pieces of information in the SPD. For its documentation, templates or structured documents are recommended, which address the contents summarized below (a sample SPD is provided in Appendix C):

1) **Configuration control information**: Required for configuration control, references to preceding IPs, and references by succeeding IPs. It should include an unique version identifier.
2) **M&S objectives hierarchy**: The M&S aims should be decomposed into a detailed list of sub-objectives with a rationale for their decomposition.
3) **Model border requirements**: Description of the required input to and output from the model.
   a) **Conceptual description**: Description of the influence (input) and goal (output) parameters, as if created or measured in the real experiment. This includes
      i) their assignments to real world measurands, for explanation of their semantics;
      ii) the explanation of assumed or postulated relevance of each parameter for the model, as a rationale for modeling particular properties;
      iii) the required domains (value ranges) and a rationale for their selection;
      iv) the required accuracies and a rationale for their selection;
      v) identification of externally provided (standardized) data models.
   b) **Formal specification**: Formal specification requirements of the model I/O (for example driven by the need for formal integration, as required by the HLA rules).
   c) **Technical implementation**: Describes interfaces and protocols for the exchange of electronic I/O data with data bases, other models, or other data sources and data recording tools. This affects
      i) the format of input (standards of data repositories, interfaces, input generation tools) and the format of output (standards for data collection) of the overall model;
      ii) interface and protocol specifications for xIL, including the rationale for their selection and response time requirements;
      iii) communication infrastructure and a rationale for its selection;
      iv) simulation libraries or simulation environments (API, support functions, library description) and rationale for their selection;
      v) real time requirements (maximum allowed response times).
4) **Model structure and behavior requirements**: Description of a priori known subsystems and their behavior that shall be considered during modeling. For non-closed, non-stand alone models, this might be a requirement for substantive interoperability with other models or systems.
   a) **Conceptual description**:
      i) A priori identified structural elements: This includes real subsystems relevant to the intended purpose, with explanation of their relevance and their allocation within the composition hierarchy of the real system, for explanation of their semantics; description of their behavior and identification of involved input and affected output parameters of the next higher submodel; description of their border, conceptual I/O requirements and constraints;

ii) Imported structural elements: This addresses the above for imported conceptual submodels, software modules, hardware subsystems, or human operators;
iii) Data models: Data models that the structure of the CM must be compliant with, with (a reference to) the description of the data model and an explanation of the necessity of compliance;
iv) A priori identified behavior: Already known behavior sequences or interactions between a priory identified or imported structural elements, including assignment to real world behavior (for explanation of their semantics), and explanation of their relevance (as rationale for their introduction), or commonly accepted functional descriptions (e.g., Functional Descriptions of the Mission Space, FDMS [Defense Modeling and Simulation Office 2003]);
v) A priori identified behavioral constraints: Specification of behavior that must (not) be shown by the model or its submodels without knowing or explicitly postulating the exact functional dependencies, with assignment to real world behavior (for explanation of their semantics), and an explanation of their relevance (as a rationale for their introduction).

b) **Formal specification**: Identification and (a reference to) the description of the required modeling formalism(s) (e.g. QN, PN, DEVS), templates for model documentation (e.g., HLA OMT), or other formal specification methods (e.g., UML, Z).
c) **Technical implementation**:
i) Software requirements: Simulation environment, programming paradigm, programming language, and plug-ins, operating system, in parallel running processes (e.g., Runtime Communication Infrastructure, Object Request Broker, other submodels, or SIL), and the rationale for their selection.
ii) Hardware requirements: I/O devices, HIL, execution platform, and the rationale for their selection.

5) **Experimental framework requirements**: General type of scenarios that shall be examined using the model.
a) **Conceptual setup**: Scenarios described as sample experiments for achieving the identified sub-aims, input data with accuracy requirements, including reference scenarios with predicted outcome the simulation must duplicate;
b) **Technical setup**: Describes data base access, instrumentation, and output recording.
6) **Additional acceptability criteria**: Identification of all additional requirements that the model and simulation results must satisfy to be acceptable. As a minimum for acceptance all requirements stated in the SPD need to be satisfied.
7) **Standards summary**: All standards that shall be considered during model development and use.

All entries without information have to be justified. The required information may be completed during several iterations of the problem definition phase.

2.3 Potential Errors in the SPD

The SPD documents the results of the problem definition phase and is the foundation for all further model development. The dependencies between the identified failures during problem definition and errors in the SPD are sketched in Figure 1. The following errors should be detected as early as possible:

- **Incorrectness in form** (internal error): Relevant types of information for the development of the model are not addressed (incompleteness; e.g., no documentation of influence parameters is found). Forgetting any type of information opens undesired degrees of freedom for modeling and most probably results in an unsuitable realization of the model (e.g., inadequate system border, unsuitable degree of abstraction, imple-

mentation for the wrong computer platform). Incorrectness in form is also caused by the violation of format conventions. It can only be determined, if there is a clearly identified *reference form* (as presented in section 2.2) that describes the required contents of the SPD.

- **Internal inconsistency** (internal error): The contents of the SPD are contradictory. (For example, imported structural elements are not compliant with the required data model, or the required model I/O does not allow the execution of the required experiments.) The requirements concerning concept, formalization, implementation, and execution with any logical dependency must be consistent. An *internal dependency matrix* can be created from the same reference (checklist, template) that is used for the detection of incorrectness in form.
- **Ambiguity** (internal error): The contents of the SPD leave too much freedom for interpretation.
- **Incorrectness in fact** (internal error): The SPD is wrong with respect to available system knowledge, domain knowledge, or metrics definition, independent of the task that shall be accomplished by M&S.
- **Incompleteness in content** (transformation error): Relevant information for the development of the model is missing (e.g., an important influence parameter or parameter of interest was forgotten).
- **Inconsistency with SN** (transformation error): The SN are not suitably reflected by the SPD. Even, if all required types of information have been considered (correctness in form) and are self-consistent, the information itself may be wrong, e.g., an unsuitable real system was identified as a reference system.

The importance of a precise and complete requirements specification for model development is also stressed in [Balci 2000]. [Partsch 1991] discusses requirements specification for the development of software in general. Due to the nature of modeling, suitability can only be judged with respect to the intended purpose, which ideally is documented in form of an M&S objectives hierarchy. The success of a validation attempt of a model or simulation results strongly depends on the quality of the definition of the intended purposes.
The following V&V activities are performed to demonstrate the absence of errors in the SPD.

2.4 SPD Internal V&V

This section documents the goals of V&V sub-phase 1.1, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 2 visualizes their dependencies.

- **Required stage of model development**: Problem definition phase completed
- **IP counterpart for comparison** (transformation V&V only): N/A
- **Sub-phase goal**: Demonstrate formal correctness and self-consistency of SPD. This sub-phase aims to ensure that a well-documented, complete, internally consistent, factually correct, and reasonable SPD is available. Addressed errors include that
  - the M&S aims hierarchy,
  - the identification of the real system,
  - the conceptual modeling requirements,
  - formalization requirements,
  - implementation requirements, and
  - experimental requirements

  are missing, imprecise, incorrect, inconsistent with other entries, or not testable.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. Errors in the SPD may propagate through the complete model

development process and cause serious trouble. Missing requirements open undesired degrees of freedom, while factually incorrect requirements are most likely to result in a factually incorrect model.

- **Impact of error detection on model development**: Regress by one phase. The problem definition phase has to be (at least partially) repeated.
- **Sub-phase objectives**: Demonstrate that
  - the SPD is correct in form (i.e., consistent with the required SPD template);
  - all contents of the SPD are consistent;
  - all contents of the SPD are factually correct (i.e., consistent with domain knowledge);
  - all contents of the SPD are unambiguous;
  - all contents of the SPD are testable.

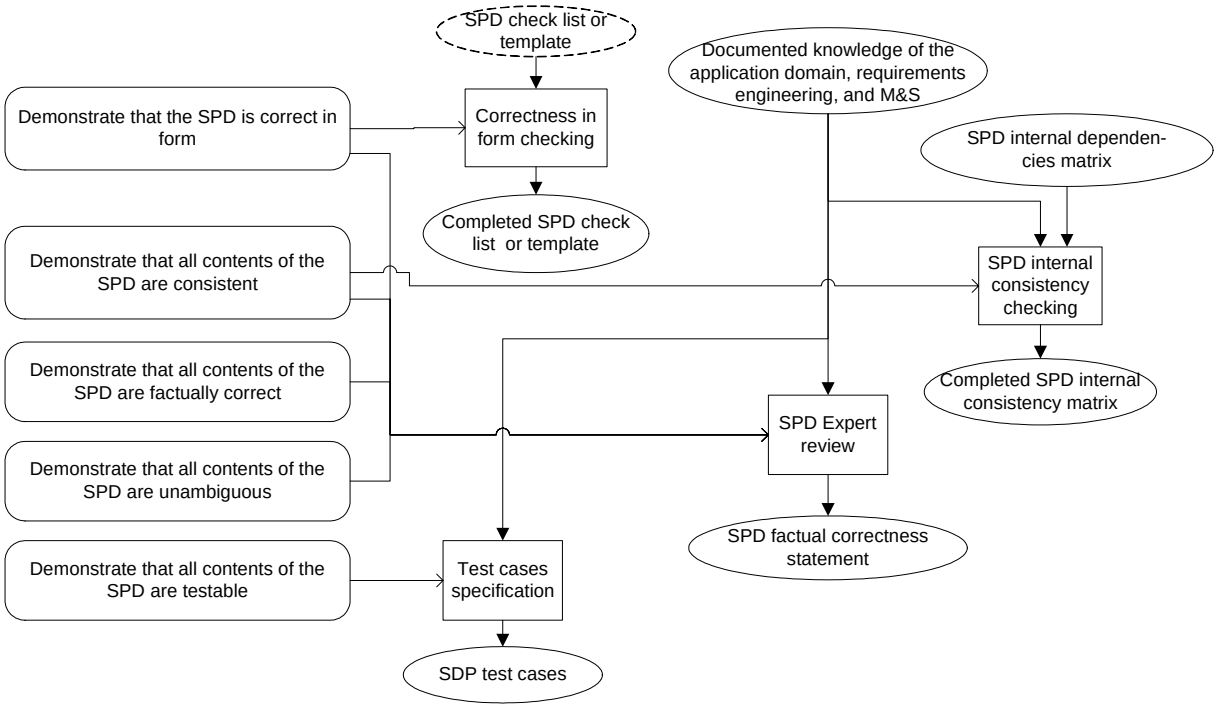


Figure 2: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 1.1

- **Proposed techniques for static analysis**:
  - SPD documentation correctness in form checking (includes completeness checking): Using an appropriate checklist or a template it is checked, whether the information required by the checklist or the fields of the template is provided in the required format. This results in a completed SPD checklist or template.
  - SPD documentation internal consistency checking: Using a template-based internal dependency matrix, which identifies the contents of the SPD that need to be consistent, they are judged for consistency by an expert. The outcome is documented in a completed SPD internal consistency matrix.
  - SPD expert review: The contents of the SPD are evaluated for consistency with domain knowledge, requirements engineering knowledge, and M&S knowledge. This results in an SPD factual correctness statement.
  - Test case specification: For all (testable) contents of the SPD, test cases are outlined. The test cases specification include expected test outcome (oracles) and pass/fail criteria.

- **Proposed techniques for dynamic testing**: N/A
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
    - SPD checklist or template
    - SPD internal dependency matrix
    - Documented knowledge of the application domain, requirements engineering, and M&S
- **Provided output**:
    - Completed SPD checklist or template, including correctness statement
    - Completed SPD internal consistency matrix
    - SPD factual correctness statement
    - SDP test cases

2.5 SN to SPD Transformation V&V

This section documents the goals of V&V sub-phase 1.2, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 3 visualizes their dependencies.

- **Required stage of model development**: Problem definition phase completed
- **IP for comparison** (transformation V&V only): SN
- **Sub-phase goal**: Demonstrate completeness of the SPD in content and consistency with the SN. This sub-phase aims to ensure that the SPD indeed reflects the SN as precisely as required. Addressed errors include:
    - Incorrect or unsuitable M&S objectives: The achievement of the specified M&S objectives does not contribute to the satisfaction of the SN.
    - Incorrect or unsuitable conceptual modeling requirements: The consideration of the conceptual modeling requirements does not lead to the needed representation of the real system.
    - Incorrect or unsuitable real reference system: The specification of the reference system does not support the identification of a sufficiently similar real system.
    - Incorrect or unsuitable formal modeling requirements: The consideration of the formal modeling requirements does not result in the needed formal specification.
    - Incorrect or unsuitable implementation requirements: The consideration of the implementation requirements does not yield the needed technical solution.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. Errors in the SPD may propagate through the complete model development process and cause serious trouble. If unsuitable requirements, or requirements, which are inconsistent with the SN are documented, it must be doubted, whether the overall needs will be satisfied in the end.
- **Impact of error detection on model development**: Regress by one phase. The problem definition phase has to be (at least partially) repeated.
- **Sub-phase objectives**: Demonstrate that
    - the model border (I/O parameters, accuracy, update conditions) is defined as needed;
    - all developmental constraints (conceptual modeling requirements, formalization requirements, implementation requirements) are defined as needed;
    - the experimental framework is defined as needed;
    - the specified test cases allow the assessment of acceptability criteria satisfaction.

- **Proposed techniques for static analysis**:
  - SN-SPD completeness and consistency checking: A trace is created from each explicitly stated need to the associated requirements in the SPD, and documented in an SN-SPD traceability matrix. Requirements without needs and needs without requirements need to be resolved. With a template or checklist for the SN available, a template-based SN-SPD dependency matrix can be created and used.
  - SPD expert review: An expert with knowledge of the problem (sponsor representative) reviews the SPD and assesses its completeness and correctness in content. The review statements need to be documented as an SPD adequacy statement.
  - Test case review and pass/fail criteria confirmation: It is ensured that the test cases cover all testable needs. It may become necessary to adjust them to the particular test conditions.
- **Proposed techniques for dynamic testing**: N/A
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
  - Problem knowledge beyond the available SN documentation
  - Expert knowledge: Domain knowledge, requirements engineering, M&S
  - Template-based SN-SPD dependency matrix
  - Test cases from 1.1
- **Provided output**:
  - SPD adequacy statement
  - Completed SN-SPD traceability matrix
  - Additional test cases with predicted outcomes and pass/fail criteria

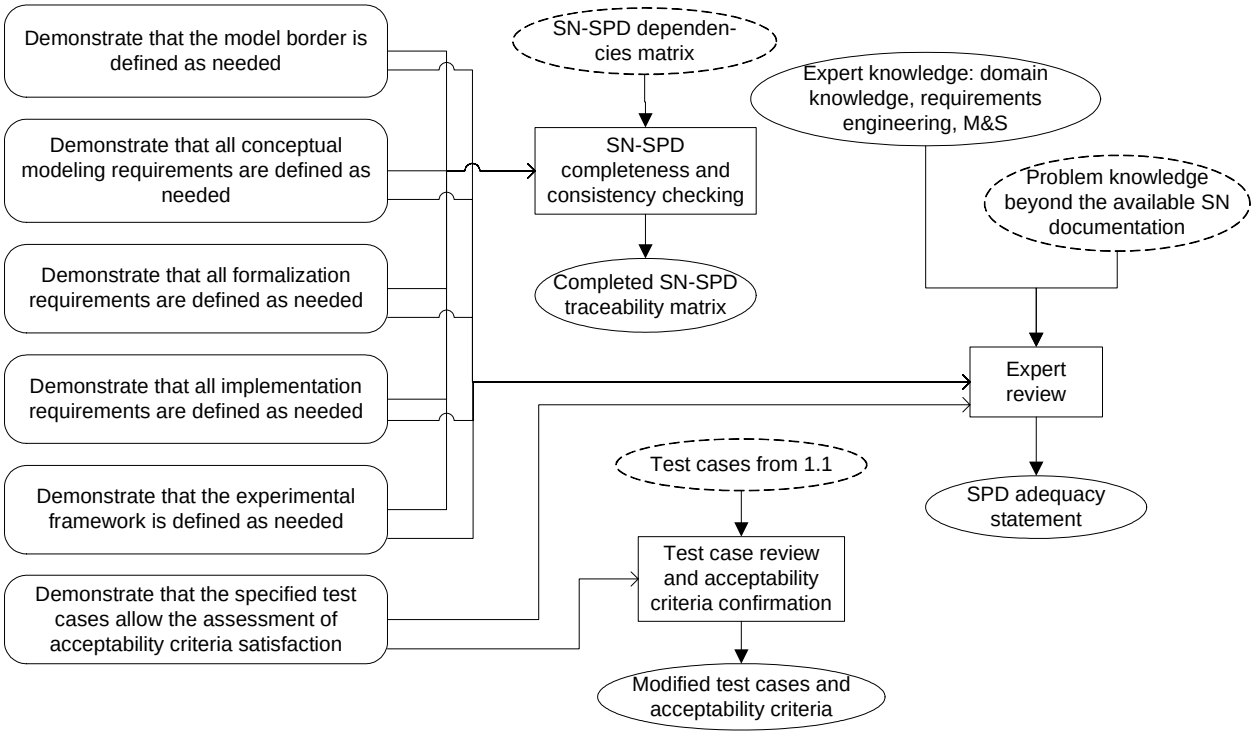


Figure 3: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 1.2

3 V&V OF THE CONCEPTUAL MODEL

The CM documents the modelers understanding of the real system in an idealized and abstract form. It serves as a comprehensive foundation for communication between those familiar with the application domain (sponsor, user, system analyst) and those responsible for the development of the model (modeler, programmer).

3.1 Expected System Analysis Activities

Goal of the second phase of the model development process is to collect as much information about the real system as required to identify all relevant subsystems, their internal behavior and inter subsystems dependencies, giving consideration to the aims, requirements, and constraints given in the SPD. As a result, the idealized and abstracted observed, perceived, or assumed real system is documented as the CM.
Failures during examination of the real system, its decomposition into subsystems, and during data collecting cause errors that affect the correctness and suitability of the information contained within the CM.

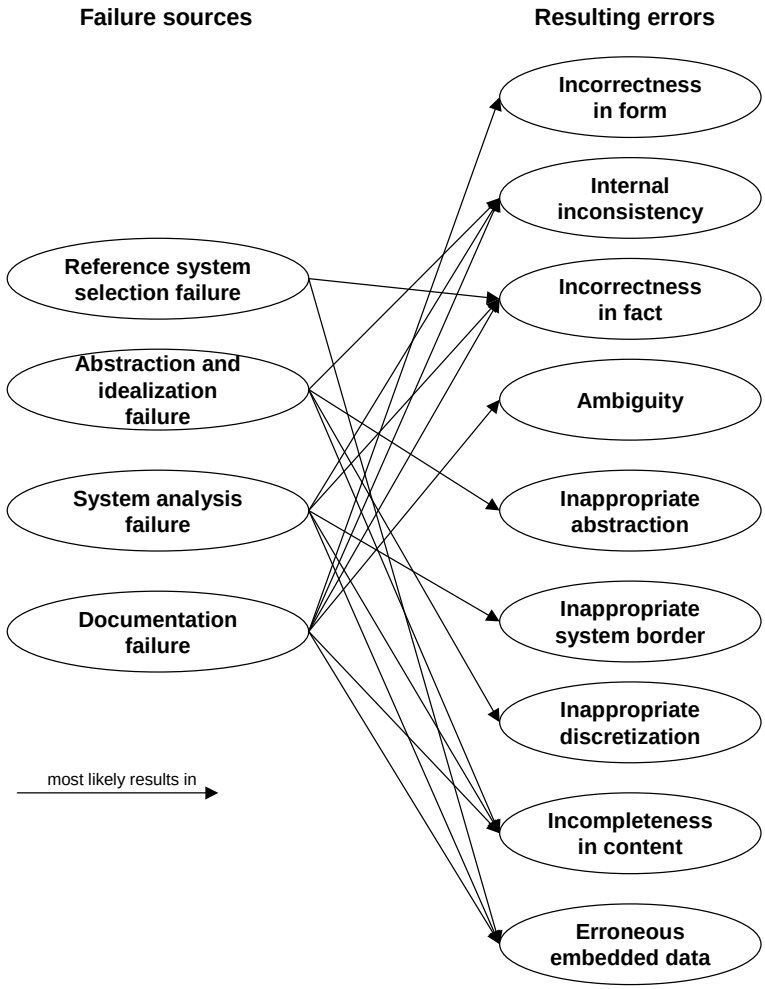


Figure 4: Dependencies between failures in the system analysis phase and errors in the CM

The CM is created in cooperation between modeler and system analyst and will be used by the modeler in the following phases of model development. Failures include superficial and insufficient system analysis, unsuitable measurement techniques, or may be due to a wrong SPD.

- **Reference system selection failure**: Whenever a model reflects a system whose behavior cannot be observed for some reason, one tries to derive the structure and behavior description from existing or easier available knowledge. This includes related systems, documented knowledge, and expert opinion. Wrong theoretical assumptions or conclusions, an SME with inadequate knowledge, or data, which are derived from a related, but not sufficiently similar real system, most likely lead to a wrong CM, including unsuitable or incorrect embedded data or runtime data.
- **Abstraction and idealization failure**: Modeling is not only a science, but also an art. One of the most challenging tasks during modeling is to choose a suitable degree of abstraction and idealization. If the model is too abstract, the lack of detail does not allow to reproduce the behavior of the real system with the desired accuracy. However, high resolution does not automatically imply better behavior representation, as for a complex system, the whole is more than the sum of its parts. The behavior of the system may be well known at a higher level of abstraction, while the knowledge about the modeled detailed interaction of its components is still immature. (For example, the behavior of a specific individual in a particular situation may be unpredictable, in contrast to the behavior of a mass of people in the same situation.) Theoretically, the number of properties of a real system that can be analyzed is unlimited, therefore during modeling under explicit consideration of the intended purpose a limited set of relevant properties is identified. Failure to do so results in an incorrect or unsuitable CM.
- **System analysis failure**: Numerous approaches to system analysis exist, including inductive and deductive techniques, top-dwn, or bottom-up approaches [Simon 1962]. These approaches assume that modeling includes creating a *subsystems hierarchy*. Each submodel corresponds to a real subsystem that has been identified, being composed of other interacting submodels on the next lower hierarchy level, as suitable. If system analysis fails, the extracted subsystems hierarchy is unsuitable or incorrect.
- **Documentation failure**: While documenting the CM, it should be assured that the suitable subspaces of the three-dimensional model information space (main document, section 4.3.1) are covered. Incomplete CM documentation is not only an obstacle to V&V, but also to model reuse.

The system analysis phase is re-entered, if the SPD is refined (this occurs frequently, as complete specification is rarely available), the SPD significantly changes, or an error in the CM is detected.

3.2 CM Documentation Requirements

The CM can be considered as (main) part of the natural language specification of the simulation software that will be developed later during the implementation phase. The representation forms of a CM are numerous; recommended are *block diagrams*, *dependency graphs*, *hierarchy trees*, *structured documents*, or *tables*. Although the CM should be self-explaining, the ambiguity of natural language implies that usually additional explanation (*presentation, discussion, explanatory text*) is needed. The CM should address the following contents (a sample CM is given in Appendix C):

1) **Configuration control information**: Required for configuration control, references to preceding IPs, and references by succeeding IPs. It should include an unique version identifier.
2) **CM overview**: A brief description of the CM, summarizing model purpose, CM border, and its contents.

3) **Structure and behavior description**: Layer-wise description of all submodels of each hierarchy layer, their internal behavior, their decomposition into lower-level submodels, and their interactions. This includes *for each submodel on each layer*:
   a) **Real subsystem**: Identification of the real subsystem reflected by the submodel.
   b) **Origin**: Identifies the submodel as custom-made for the M&S objectives documented in the (referenced) SPD, as externally provided and integrated CM originally developed for another purpose, or as CM of a formally specified or already implemented executable submodel. If a pre-defined submodel is used, a rationale for integrating it is required (e.g. identification of an interoperability standard or data model that demands its use, or assumed advantage of using it).
   c) **Submodel border**: Description of all I/O parameters, including
      i) an identifier, the specification of its range, unit, accuracy (decimal places), update frequency or condition, and initial value;
      ii) assignment to real world measurand (for explanation of its semantics);
      iii) identification of any interoperability framework or standard that the selection of I/O parameters is compliant with.
   d) **Structure description**:
      i) Detailed description of the abstract and idealized replication of the real subsystem, rationale for the chosen abstraction, and identification of the real world references used (e.g., map, CAD drawing);
      ii) Identification and rough description all of contained submodels, rationale for the chosen (de-)composition, and assignment between submodels and real subsystems (to explain of the semantics of each submodel);
      iii) Inheritance hierarchies ("is a" relationships), including rationale for choosing the hierarchies, and motivation for refinement or abstraction of modeled elements;
      iv) Identification of any interoperability framework or standard that the structure definition is compliant with.
   e) **Internal behavior description**:
      i) Identification of states or subsets of the state space: This includes the description of state variables (identifier, range set, unit, accuracy (decimal places), and initial value) and states (sets of state variable ranges), and an assignment to states of the real subsystem (to explain of the semantics of each state);
      ii) (Graphical) description of behavior or state transitions (internal behavior, as reactions to external stimuli, events, or other submodels state constellations; own internal dynamics). This includes state transitions or updates of state and output variables (state transition diagram), conditions for state transitions or variable updates, and an assignment to real state changes, to explain the semantics of each state change. Explanation for the chosen resolution of time;
      iii) Identification of any interoperability framework or standard that the behavior definition is compliant with.
   f) **Interaction with other submodels**:
      i) Identification of all submodels on the same layer in the submodel hierarchy that are interaction partners, including identification and description of interconnection between their I/O parameters.
      ii) Assignment to real world interactions, for an explanation of the semantics of each interaction.
      iii) Rationale for the derivation of conditions that initiate the interaction, referring to real world references.
      iv) Behavior constraints of the submodels that later refinement must satisfy, including illegal attribute value interval combinations or cause-effect pairs.

v) Identification of any interoperability framework or standard that the interaction definition is compliant with.

g) **Embedded data**: Reference to exhaustive data documentation.

4) **Reference material summary**: Clear identification of all reference material that is used for system analysis, e.g. CAD sketches, construction plans, documented expert opinion, data files, maps, or photos.
5) **Standards summary**: Identification of any standard the conceptual submodel is compliant with.

For each item no information is given for, there should be a justification. The required information may be completed during several iterations of the system analysis phase.

3.3 Potential Errors in the CM

This section summarizes errors that origin from incorrect or unsuitable abstraction, idealization, selection of the system border, or simply from insufficient knowledge about the real system, and result in a CM not suitable for the intended purpose or incorrect. Errors in the CM occur during system analysis and the aim-oriented identification, abstraction and idealization of, e.g., structures, objects, states, and behavior, or are the result of specification errors. The dependencies between the identified failures during system analysis and errors in the CM are sketched in Figure 4. Potential errors identified for the CM are introduced in the following:

- **Incorrectness in form** (internal error): When modeling a real system, it usually is required to identify inputs that have to be considered, outputs that should be observable, eventually giving consideration to objects or subsystems, their attributes, and their interactions, of cause-effect-dependencies, states, and state transitions, and more. If any of these types of information is missing, the CM is incomplete in form. For the judgment of correctness in form there must be a clearly identified reference form summarizing the mandatory contents of the CM (e.g. in section 3.2).
- **Internal inconsistency** (internal error): Discrepancies within the CM. Whenever there are logical dependencies between particular contents of the CM, these contents should be consistent. For example, parameters used in the behavior description of a submodel should also be defined within its structure description. A consistency checklist can be created from the same reference that is used for judgment of correctness in form.
- **Incorrectness in fact** (internal error): Violation of the rules and laws of the application domain. In each application domain certain principles and rules are valid, e.g., when modeling movement, kinetics must be considered.
- **Ambiguity** (internal error): The specification of the CM is imprecise and opens undesired degrees of freedom for further model development.
- **Unsuitable abstraction and idealization** (transformation error): To allow the achievement of the intended purpose, a suitable level of abstraction is required, including a suitable identification of subsystems and atomic subsystems. In addition, unsuitable abstraction and idealization may result in incompliance with externally provided reference models like the Functional Descriptions of the Mission Space (FDMS [Defense Modeling and Simulation Office 2003]). Compatible abstraction and idealization is required, otherwise substantive interoperability between submodels is most likely not achievable.
- **Unsuitable model border** (transformation error): Required external influences on the real system are not properly reflected by the input parameters of the model, or other relevant subsystems that influence the behavior of the subsystem of interest are not included in the model. The model output parameters do not allow the observations one is interested in.

- **Unsuitable discretization of time** (transformation error): The resolution of time or the discretization of the state space is not precise enough to allow the desired observations, or too precise for (efficient) execution of the implementation. Especially when modeling continuous processes, special care is required.
- **Incompleteness in content** (transformation error): The contents of the SPD are not satisfied, e.g., required I/O parameters or anticipated subsystems are not considered. Consistency or compliance requirements have been violated.
- **Unsuitable or incorrect embedded data** (transformation error): If data that is embedded in the model is measured at an unsuitable related system, is created by another invalid model, or is otherwise determined in the wrong way, the quantitative aspects of the behavior description are wrong. Even if there is data directly measured at the real system available, it must be ensured that it is suitable (e.g., age, measurement tolerance).

In the following V&V activities are identified to demonstrate the absence of the errors mentioned above. As conceptual modeling is not only a science, but also an art, it is extremely difficult to distinguish unsuitable from suitable at this stage of model development.

3.4 CM Internal V&V

This section documents the goals of V&V sub-phase 2.1, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 5 visualizes their dependencies.

- **Required stage of model development**: System analysis completed
- **IP counterpart** (transformation V&V only): N/A
- **Sub-phase goal**: Demonstrate formal correctness and self-consistency of the CM. This sub-phase aims mainly to ensure that the documentation of the CM is complete, internally consistent, and useful, and that the CM is consistent with the available domain knowledge. Addressed errors include that
    - the submodel border descriptions,
    - the conceptual submodel hierarchy,
    - the submodel internal behavior descriptions, and
    - the submodel interactions

  are missing, imprecise, factually incorrect, or inconsistent with respect to other CM contents.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. Inconsistencies in the CM will delay further model development, or lead to wrong simulation results. If factually wrong model contents (inconsistent with domain knowledge) are not detected in this early stage of model development, resources are wasted developing an (at least partially) factually wrong FM.
- **Impact of error detection on model development**: Regress by on phase. The system analysis phase must be (at least partially) repeated giving consideration to the newly gained knowledge about the real system.
- **Sub-phase objectives**: Demonstrate that
    - the CM (including its supplemental information and the conceptually described experimental framework) is correct in form (i.e., consistent with the required CM template),
    - all contents of the CM are consistent with each other,
    - all contents of the CM (including structure, embedded data, anticipated behavior, and experimental framework) are unambiguous and factually correct (consistent with domain knowledge),

- the data sources are suitable, and
- all data are sufficiently accurate.

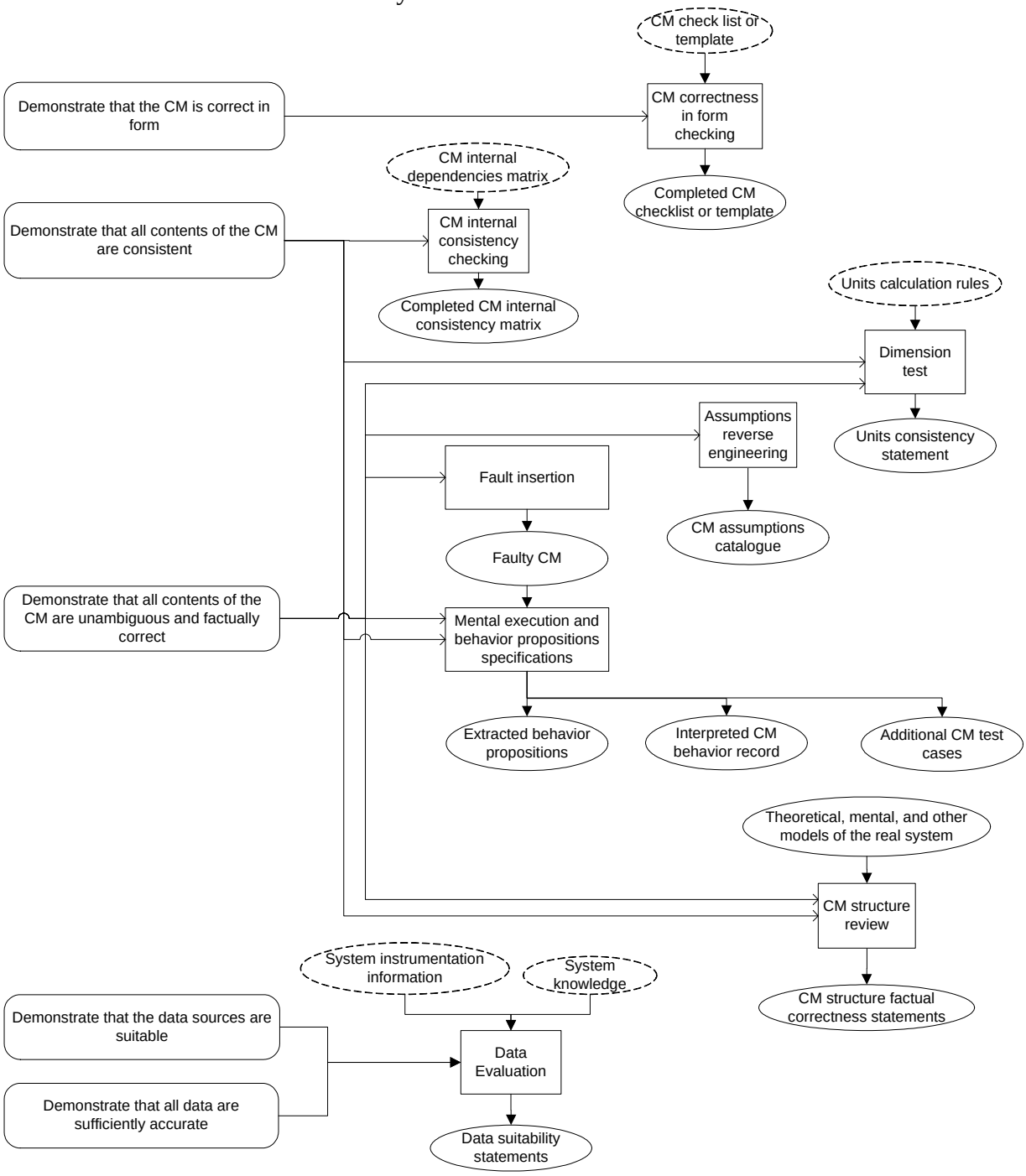


Figure 5: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 2.1

- **Proposed techniques for static analysis**:
  - CM documentation correctness in form checking (includes completeness checking): Using an appropriate checklist or a template is checked (as provided in section 3.2), whether the information required by the checklist or the fields of the template is provided in the required format. This yields a completed CM checklist or template.
  - CM documentation internal consistency checking: Using a template-based internal dependency matrix, which identifies the contents of the CM that need to

be consistent, these contents are judged for consistency by an expert. The outcome of this activity is a completed CM internal consistency matrix.
    - CM structure review: The structure description of all submodels is reviewed and assessed for consistency with available domain knowledge. This results in a CM factual correctness statement.
    - Dimension test: This test is highly specialized to check equations containing parameters with units. The resolution of the units according to the equation must be correct.
    - Assumptions reverse-engineering: The CM is reviewed by SMEs who write down the assumptions, which must hold for the model to be a valid representation of the real system. These reverse-engineered assumptions can later be used for comparison with previously documented assumptions.
    - Application domain dependent analysis and checks: Dependent on the application domain, highly specialized V&V activities can be performed. For example, for integrated circuit design the distances between different types of semiconductor materials can be verified efficiently.
- **Proposed techniques for dynamic testing**:
    - Mental execution and behavior propositions specification: The conceptual structure and behavior description of each submodel is mentally interpreted, to anticipate the behavior of the model under particular input conditions and to extract behavior propositions. Simply by reading the structure and behavior specification carefully and rigorously for the purpose of behavior propositions specification, some of its ambiguities, gaps, and loopholes already become obvious. The extracted behavior propositions can be reused for numerous V&V tasks, as, e.g., model checking of the FM, or test case generation for the EM. (Behavior propositions are the foundation of the "path of behavior specification violation detection", as explained in detail in the main document, section 5.3.) However, mental execution is cumbersome and error-prone. It usually provides only vague results for an extremely limited number of test cases.
    - Fault/failure insertion: Alternatively to the mental execution of the unmodified CM, it can be modified, e.g. by fault insertion. The (now erroneous) anticipated model behavior (failure) should be plausible. The anticipated model behavior is documented.
    - Data evaluation: Experts assess the suitability of the data used as a foundation for conceptual modeling, yielding a data suitability statement.
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
    - Theoretical, mental, and other models of the real system (system knowledge)
    - Test cases from 1.1
    - CM checklist or template
    - CM internal dependency matrix
    - Units calculation rules
    - System instrumentation information
- **Provided output**:
    - Completed CM checklist or template
    - Completed CM internal consistency matrix
    - Extracted behavior propositions for each submodel
    - Additional CM test cases
    - Interpreted CM behavior record
    - CM structure factual correctness statements
    - CM assumptions catalogue

- Data suitability statements

3.5 SPD to CM transformation V&V

This section documents the goals of V&V sub-phase 2.2, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 6 visualizes their dependencies.

- **Required stage of model development**: System analysis completed
- **IP counterpart** (transformation V&V only): SPD
- **Sub-phase goal**: Demonstrate completeness of the CM in content and consistency with the SPD. This sub-phase aims mainly to ensure that the CM meets the conceptual modeling requirements and is a correct and suitable representation of the real system, which allows the execution of the required experiments within the specified experimental framework. Explicitly addressed errors include that
  - the model border (I/O parameters, their semantics, their accuracy),
  - the conceptual submodel structure description,
  - the anticipated submodel internal behavior,
  - the anticipated submodel interactions,
  - the chosen experimental framework, and
  - the representation of time does not allow to achieve the specified M&S objectives; and that
  - explicit conceptual modeling requirements are violated (e.g., required I/O parameters, submodel structure, or submodel behavior).
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. If the CM does not meet the requirements for conceptual modeling, resources will be wasted for the development of an at least partially wrong FM.
- **Impact of error detection on model development**: Regress by one phase. The system analysis phase has to be (at least partially) repeated under special consideration of the detected inconsistencies and missing contents.
- **Sub-phase objectives**: Demonstrate that
  - the model border (I/O parameters, their semantics, their accuracy),
  - the degree of abstraction and idealization,
  - the model structure and hierarchical decomposition,
  - the representation of time,
  - the anticipated model behavior (including embedded data), and
  - the experimental framework

  meet the requirements.
- **Proposed techniques for static analysis**:
  - SPD-CM completeness and consistency checking: A trace is created from each applicable structural and behavioral requirement in the SPD to the associated content of the CM and documented in an SPD-CM traceability matrix. Contents without requirements and requirements without model contents need to be resolved. A template-based SN-SPD dependency matrix can be created and used for this purpose.
  - CM assumptions review: The assumptions catalogue created during V&V sub-phase 2.1 is reviewed for its consistency with the SPD. If the extracted assumptions contradict the conditions under which the model is supposed to be exercised to accomplish the required M&S objectives, a problem is detected. The results of the review are documented in the CM assumptions compliance statement.

- CM structure review: The symbolic structure description is reviewed for a coverage of all structural elements required in the SPD. The results of the review are documented in the CM structure compliance statement.
- Behavior propositions review: The behavior propositions extracted from the CM during V&V sub-phase 2.1 are compared to the behavior requirements given in the SPD. If there are behavior requirements without associated behavior propositions (and vice versa), or if required behavior and behavior propositions are inconsistent, a problem is detected. This is documented in a CM behavior propositions compliance statement.

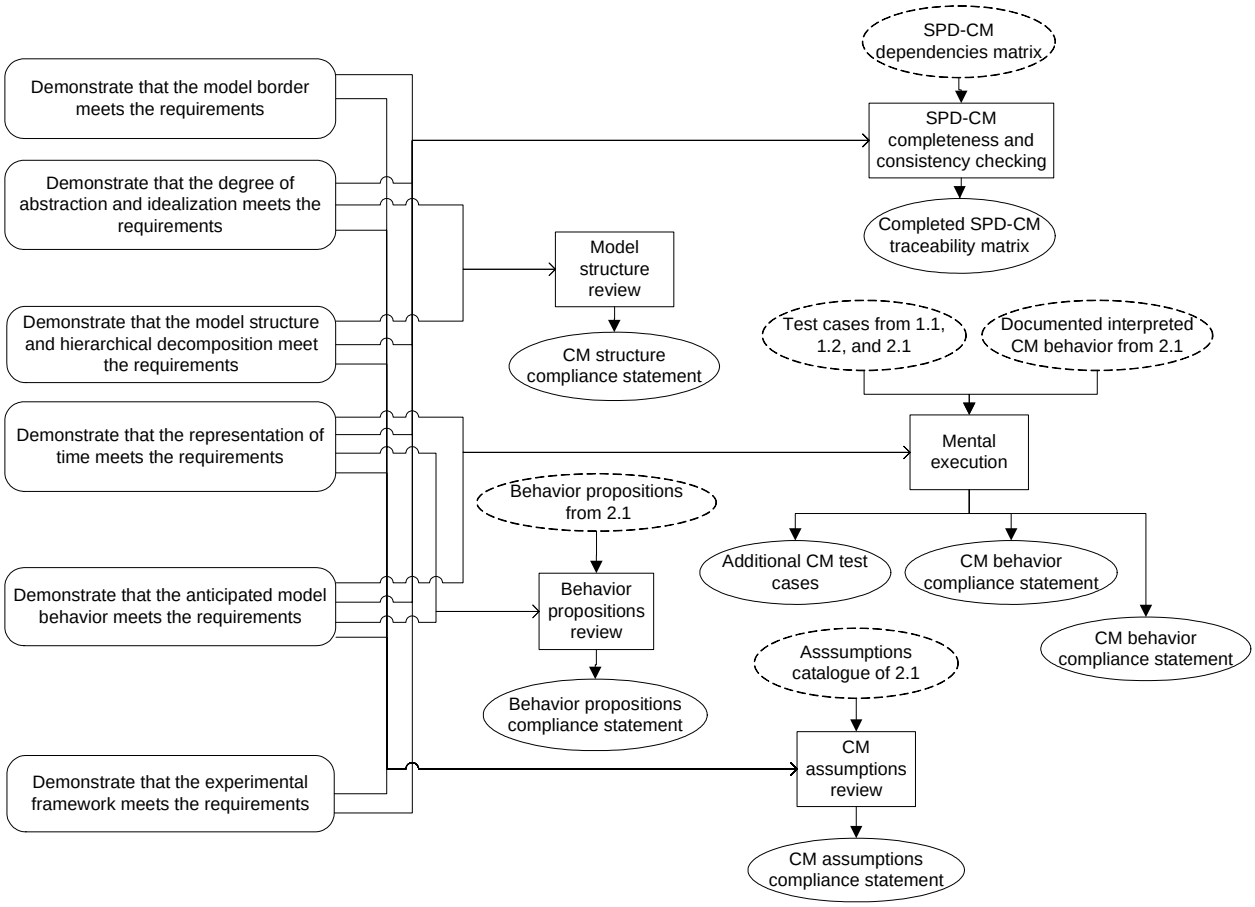


Figure 6: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 2.2

- **Proposed techniques for dynamic testing**:
  - Mental execution: A subset of the test cases identified during the V&V sub-phases 1.1, 1.2, and 2.1 is mentally executed to anticipate CM behavior, which supports judgment of consistency between the symbolic behavior description stated in the CM and the behavior requirements in the SPD. If other test cases are made up in addition to those already available, new interpreted CM behavior ("oracles") and behavior propositions should be added to the CM test cases and the CM behavior propositions, respectively.
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
  - Template-based SPD-CM dependency matrix
  - Behavior propositions from 2.1

    - Test cases from 1.1, 1.2, and 2.1
    - Interpreted CM behavior ("oracles") from 2.1
    - Assumptions catalogue from 2.1
- **Provided output**:
    - Completed SPD-CM traceability matrix
    - CM structure compliance statement
    - Additional CM test cases
    - Additional CM behavior propositions
    - CM behavior compliance statement
    - CM assumptions compliance statement

3.6 SN to CM transformation V&V

This section documents the goals of V&V sub-phase 2.3, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 7 visualizes their dependencies.

- **Required stage of model development**: System analysis completed
- **IP counterpart** (transformation V&V only): SN
- **Sub-phase goal**: Confirm completeness of the CM in content and consistency with the SN. This sub-phase mainly aims to countercheck that the SN (from which the requirements originate) are covered by the CM. With the CM now serving as a vehicle for communication, additional, in sub-phase 1.2 unavailable options are given. Typically addressed errors include that
    - an unsuitable system border does not allow to observe the needed influences and output parameters with the needed semantics,
    - the conceptual submodel structure does not allow the observation of the needed submodel interactions or internal submodel behavior (state parameters), and
    - the anticipated conceptual submodel internal behavior is too imprecise to allow the needed insight into the subsystem behavior.
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. It can be skipped, if during V&V sub-phase 1.2 it was made credible that all relevant aspects of the SN have been included suitably in the SPD, and during V&V sub-phase 2.2 it was made credible, that the CM satisfies all SPD contents. However, as it is extremely difficult to capture all M&S requirements in the SPD (truly complete specifications are rare), it is highly probable that the user's abstracted and idealized perception of the real system and the experimental framework deviates from the modeler's. As the CM is the foundation for formalization and implementation, especially during these early phases of model development direct feedback of the sponsor is crucial for success.
- **Impact of error detection on model development**: Regress by two phases. The model developer has to return to the problem definition phase, restart model development with the newly gained knowledge about the (partial) incorrectness or unsuitability of the CM or SPD, and correct the errors made during problem definition or system analysis.
- **Sub-phase objectives**: Confirm that
    - the model border (including anticipated accuracy of parameters),
    - the degree of abstraction,
    - the model structure and hierarchical decomposition,
    - the representation of time,
    - the model behavior (including embedded data), and

- the experimental framework

allow the needed experimentation. Remark: The objectives are the same as those of V&V sub-phase 2.2, with the sole exception that not the compliance with the SPD is assessed, but the adequacy with respect to the real needs.

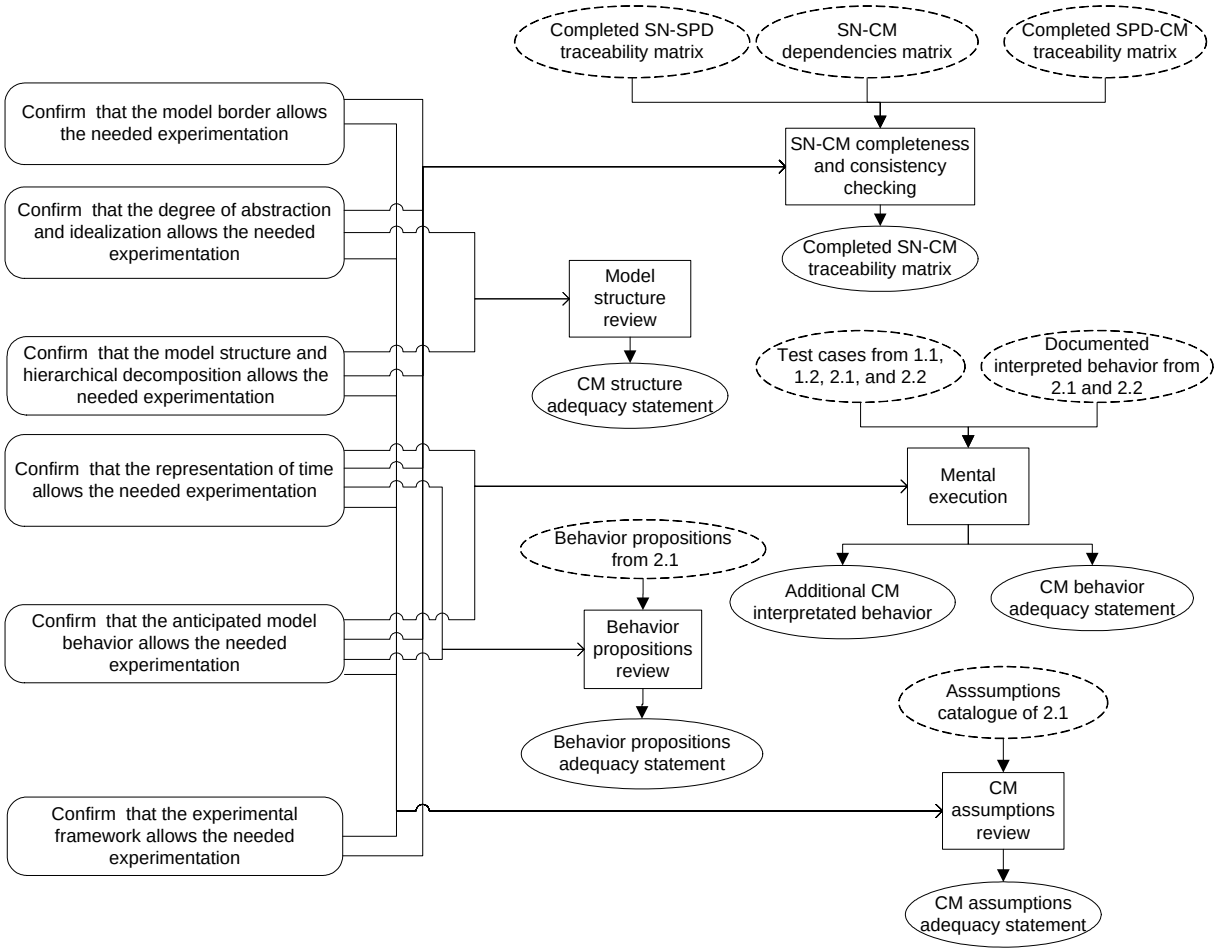


Figure 7: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 2.3

- **Proposed techniques for static analysis**:
  - SN-CM completeness and consistency checking: By reusing the SN-SPD and SPD-CM consistency matrices from V&V sub-phases 1.2 and 2.2, and the use of a template based SN-CM dependency matrix, a trace from each need to the associated CM contents is created, and documented in the SN-CM traceability matrix. Needs without associated CM contents (and vice versa) indicate a problem.
  - CM assumptions review: The assumptions catalogue created during V&V sub-phase 2.1 is reviewed by a sponsor representative with knowledge of the sponsor's needs. If the previously extracted assumptions contradict the conditions under which the model needs to be exercised to address the SN, a problem is detected. The results of the review are documented in the CM assumptions adequacy statement.
  - CM structure review: The symbolic structure description is reviewed for coverage of all structural elements, which need to be stimulated or observed. The results of the review are documented in the CM structure adequacy statement.

    - Behavior propositions review: The behavior propositions extracted from the CM during V&V sub-phase 2.1 and 2.2 are compared to the needed behavior. If needed behavior and behavior propositions are inconsistent, a problem is detected. This phenomenon is to be documented in the CM behavior adequacy statement.
- **Proposed techniques for dynamic testing**:
    - Mental execution: Additional mentally generated model behavior examples may be required to judge consistency between CM and SN.
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
    - Test cases from 1.1, 1.2, 2.1, and 2.2
    - Documented interpreted behavior from 2.1 and 2.2
    - Behavior propositions catalogue from 2.1
    - Assumptions catalogue from 2.1
    - Completed SPD-SN traceability matrix from 1.2
    - Completed CM-SPD traceability matrix from 2.2
    - Template-based SN-CM dependency matrix
- **Provided output**:
    - Completed CM-SN traceability matrix
    - Additional CM interpreted behavior (oracles)
    - Behavior propositions adequacy statement
    - CM structure adequacy statement
    - CM behavior adequacy statement
    - CM assumptions adequacy statement

4 V&V OF THE FORMAL MODEL

Quantitative solution of a model or mathematical-digital simulation on a digital computer require to express the contents of the CM in a computable form. This is achieved by using a specialized formalism with a well-defined mathematical foundation. The FM is the solution-oriented, platform-independent, syntactically and (partially) semantically unambiguous specification of the CM, according to one or more well-defined formalisms based on a mature mathematical foundation. Usually, modeling formalisms require that all contents of the CM are quantitatively described. Still missing quantitative information is added, and functional dependencies, which are too complex to achieve an efficient, precise solution are approximated (this may require additional abstraction and idealization). An interpreter can “run” or execute the FM, if it is completely defined, or can generate executable code from it.

4.1 Expected Formalization Activities

To allow the application of mature solution techniques, the CM is transformed into the FM. For mature modeling formalisms, application domain oriented and application domain independent simulation tools are available that allow direct automated interpretation of the model

or generation of executable code. For each submodel a suitable modeling formalisms is chosen during the formalization phase. An increase of model internal consistency is already achieved during formalization, because the modeler has to rethink the CM over in order to express it formally in a restricted and controlled language. Formalization forces the modeler to implicitly consider the expert knowledge that led to the development of the formalism, and helps to avoid dead-end roads during further modeling. Inconsistencies become obvious, and gaps or ambiguities in the CM are detected, when rigorously transforming it to a formal description. As a rule, the more specialized the formalism, the less flexible is its application, but its subsequent analytic solution, its interpretation, or the transformation into an EM become easier.

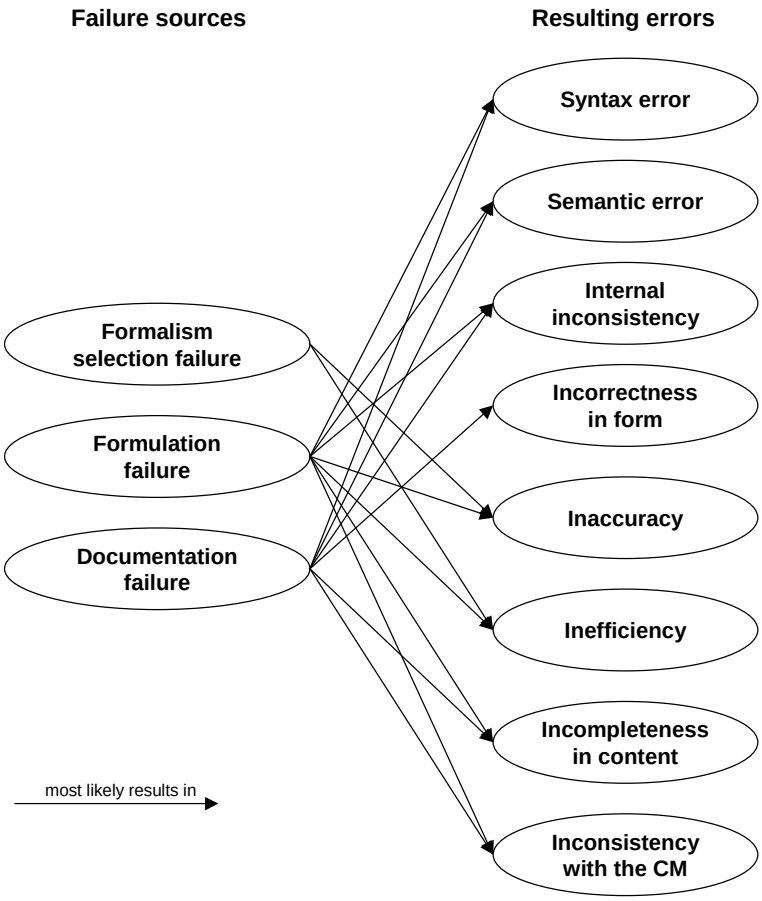


Figure 8: Dependencies between failures in the formalization phase and errors in the FM

Examples for highly specialized modeling formalisms are *Queuing Networks* (QN) for bottleneck analysis [Hillier and Liebermann 1997], or *Petri Nets* (PN) for examination of parallelisms [Lindemann 1998]. Several extensions for these "classic" formalisms are available to increase their expressiveness, including Colored QN, Extended QN, Colored PN, Timed PN, Generalized Stochastic PN, Deterministic and Stochastic PN, or Queuing PN. The challenge of hierarchical integration of submodels into a submodel hierarchy (which is not explicitly addressed by the above formalisms) for the purpose of discrete event simulation is among the key features of the Discrete Event System Specification (DEVS) of [Zeigler, Praehofer, and Kim 2000]. To allow the integration of non-discrete elements into the specification, a Discrete Time System Specification (DTSS) and a Differential Equation Systems Specification (DESS) have been formalized by Zeigler et. al. More generic (half-) formal specification techniques are more flexible, but usually provide a less strict framework for V&V. An example of a highly flexible half-formal model description method is the well known Unified

Modeling Language (UML) [Object Management Group 1999]. However, UML specifications prepare the implementation of object-oriented software, simulation specific elements are not predefined there. (In addition to the formalism mentioned here as a preparation of an approach to Multi-Formalism Modeling, [Vangheluwe 2001] provides an excellent overview of formal modeling methods.)
The following problems may occur during formalization:

- **Formalism selection failure**: For the chosen formalism no efficient solution method is available, or it is not expressive enough to allow the suitable formulation of the CM. An immature modeling formalism may contain ambiguous elements.
- **Model formulation failure**: The chosen approximation methods and numerical solution methods do not allow to express the CM with sufficient accuracy or efficiency. Also, the preparation of input data, including selection of distribution functions or fitting measured data to distribution functions, might fail.
- **FM documentation failure**: In documenting an FM, not only for the purpose of V&V it should be assured, that the suitable subspaces of the three-dimensional model information space (main document, section 4.3.1) are covered.

The model formalization phase is re-entered, when the CM changes or an error in the FM is detected. Constraints that have been considered but cannot yet be realized should be propagated through the phase.

4.2 FM Documentation Requirements

The modeler creates the FM, which can be used by the programmer as a high level design for the implementation of the EM (software). The representation form of the FM is defined by the chosen formalism and consists of, e.g., Petri Net or Queueing Net elements, DEVS equations, UML diagrams, or mathematical formulas. The formalism must have a mature mathematical foundation. Required contents are (a sample FM is given in Appendix C):

1) **Configuration control information**: Required for configuration control, references to preceding IPs, and references by succeeding IPs. It should include an unique version identifier.
2) **Identification of the formalism(s)**: This includes a reference to the description of the underlying formalism, and a rationale for its selection.
3) **Structure and behavior description**: Layer-wise description of all submodels of each layer according to the modeling formalism, (e.g., graphic diagrams or mathematical formulae), their internal behavior, their decomposition into lower-level submodels, and their interactions. This includes *for each submodel on each layer*:
   a) **CM**: Identification of the associated conceptual submodel.
   b) **Origin**: Identifies the formal submodel as custom-made, or as an FM of an implemented executable submodel. If a pre-defined submodel is used, a rationale for integrating it is required (e.g. identification of the interoperability standards that demand its use, or assumed advantage of using it).
   c) **Model border description**: According to the modeling formalism,
      i) formalized description of input data sources (model structure description),
      ii) formally correct integration of submodels, including submodels specified in different modeling formalisms.
   d) **Structure and internal behavior description**: According to the formalism, including
      i) submodel hierarchy,
      ii) static interdependencies between the submodels,
      iii) object inheritance hierarchy,
      iv) states and state transition diagrams,

v) algorithms, and heuristic and numeric solution methods explanation, including a detailed description of chosen solution methods and justification for their selection and an approximation of error, and
vi) formalized behavior constraints (assertions specification).

e) **Submodel interactions**:
i) external events and conditions for their generation
ii) formalized dynamic interactions between submodels

4) **Formalized experimental framework**: A formal specification of external influences (e.g., external event generation patterns with formatted data reference)
5) **Standards summary**: An identification of any standard the formal submodel is compliant with.

There should be a justification for each number no information is given for. The required information may be completed during several iterations of the formalization phase.

4.3 Potential Errors in the FM

All violations of the rules of model formalization and formulation, including the FM documentation requirements, are summarized as formalization errors. The completeness, clarity, and correctness of the FM must be assured. Formalization errors are the results of failures in expressing the CM as an FM according to the chosen modeling formalism. The dependencies between the identified failures during formalization and errors in the FM are sketched in Figure 8. Potential errors identified for the FM are introduced in the following:

- **Syntax errors** (internal error): Violation of the syntax rules of the modeling formalism. Each formalism defines elements and rules, which regulate the allowed dependencies between the elements. To ensure the unambiguousness of the description, these rules must not be violated. The reference for syntax checking is part of the specification of the formalism.
- **Semantic errors** (internal error): This type of incorrectness addresses the violation of the semantic rules of the modeling formalism. Often is it not allowed to combine the elements given by the formalism in any desired way. Errors occur, if, e.g., different contents of the CM are projected on elements of the formalism, and these elements are not allowed to have the dependency as intended with the CM.
- **Incorrectness in Form** (internal error): A modeling formalism defines all of its elements and their allowed use syntactically and semantically, which means that most often incorrectness of the model specification in form results in a syntax or semantic error. However, the above template requires entries in addition to the pure model contents; for example, a reference to the description of the formalism or a standards summary need to be provided, too. For the assessment of correctness in form, there must be a clearly identified reference form summarizing the mandatory contents of the FM, as given in section 4.2.
- **Internal inconsistency** (internal error): The modeling formalism defines the internal consistency requirements, which means that inconsistency usually leads to a syntax or semantic error. However, consistency with other entries, for example the formalism identification, should be ensured.
- **Ambiguity** (internal error): The FM should not be ambiguous. If it is, it must be judged, whether this allows an undesired degree of freedom for implementation.
- **Incompleteness in content** (transformation error): The FM does not reflect the CM completely.

- **Inconsistency with the CM** (transformation error): The formal specification of the abstracted and idealized real system deviates from its conceptual description. The contents of the FM thereby become factually incorrect.
- **Inaccurate description** (transformation error): The CM is not precisely enough reflected. Used numeric approximation methods are not sufficiently accurate or do not converge. Although the FM is still factually correct, it lacks the required accuracy.
- **Inefficient description** (transformation error): The chosen formal specification may be not sufficiently efficient to support an implementation that, e.g., satisfies given real-time requirements.

The absence of the above errors in the FM can be demonstrated by the implementation of the following V&V activities.

4.4 FM Internal V&V

This section documents the goals of V&V sub-phase 3.1, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 9 visualizes their dependencies.

- **Required stage of model development**: Formalization phase completed
- **IP counterpart** (transformation V&V only): N/A
- **Sub-phase goal**: Demonstrate formal correctness and self-consistency of the FM. This sub-phase aims mainly to ensure that the FM and its supplemental information are well-documented and of sufficiently high quality for subsequent V&V activities. Mainly addressed errors include that the FM and its supplemental information are formally incorrect or ambiguous.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. The syntax and semantics of a modeling formalism contain many relevant aspects for the subsequent solution of the model. Further examinations of the FM and development of the EM may be significantly disturbed by syntactical and semantic errors in the FM.
- **Impact of error detection on model development**: Regress by one phase. The model formalization phase has to be (at least partially) repeated.
- **Sub-phase objectives**: Demonstrate that
  - the FM, its supplemental information, and the formally specified experimental framework are correct in form (i.e., consistent with the required FM template) and consistent with each other;
  - there are no syntax or semantic errors in the FM i.e., the FM is consistent with the modeling formalism specification;
  - there is no ambiguity in the FM.

  Remark: For the FM, demonstration of factual correctness is not explicitly required (as it is for the CM). Under the assumption that the facts knowledge of model formalization according to a particular modeling formalism is represented by the associated modeling formalism, demonstration of factual correctness is completely covered by the demonstration of formal correctness.
- **Proposed techniques for static analysis**:
  - FM documentation correctness in form checking (includes completeness checking): Using a checklist or a template as provided in section 4.2, the FM is checked for completeness and consistency with this template. For the FM this only affects the supplemental information, because the question of correctness in form of the formalized model contents is covered by the required compliance with the modeling formalism specification.

- FM documentation internal consistency checking: In using a template-based FM internal dependency matrix, the FM documentation is checked for internal consistency. This affects only the supplemental information and its consistency with the actual FM, as the internal consistency of the FM is covered by the required compliance with the modeling formalism specification.
- Syntax analysis: The FM specification is analyzed for syntactical correctness with respect to the modeling formalism specification. All symbols used for the FM and their combination must be consistent with the grammar provided by the modeling formalism. This technique is similar to syntax analysis for program code as in [Beizer 1990].
- Semantic analysis: Combinations of and operations on elements in the FM must be compliant with their definition according to the modeling formalism. This technique is similar to semantic analysis for program code as in [Beizer 1990].
- Interfaces consistency checking: Modeling formalisms that support the hierarchical integration or composition of submodels also support rigorous analysis of consistency of the formally specified interfaces.
- Formal proof of formalism dependent properties: Some modeling formalisms allow the proof of particular desired behavior properties. For example, for Petri Nets the demonstration of the absence of deadlocks ("aliveness") can be performed efficiently. The applicability of formal proofs strongly depends on the chosen modeling formalism.

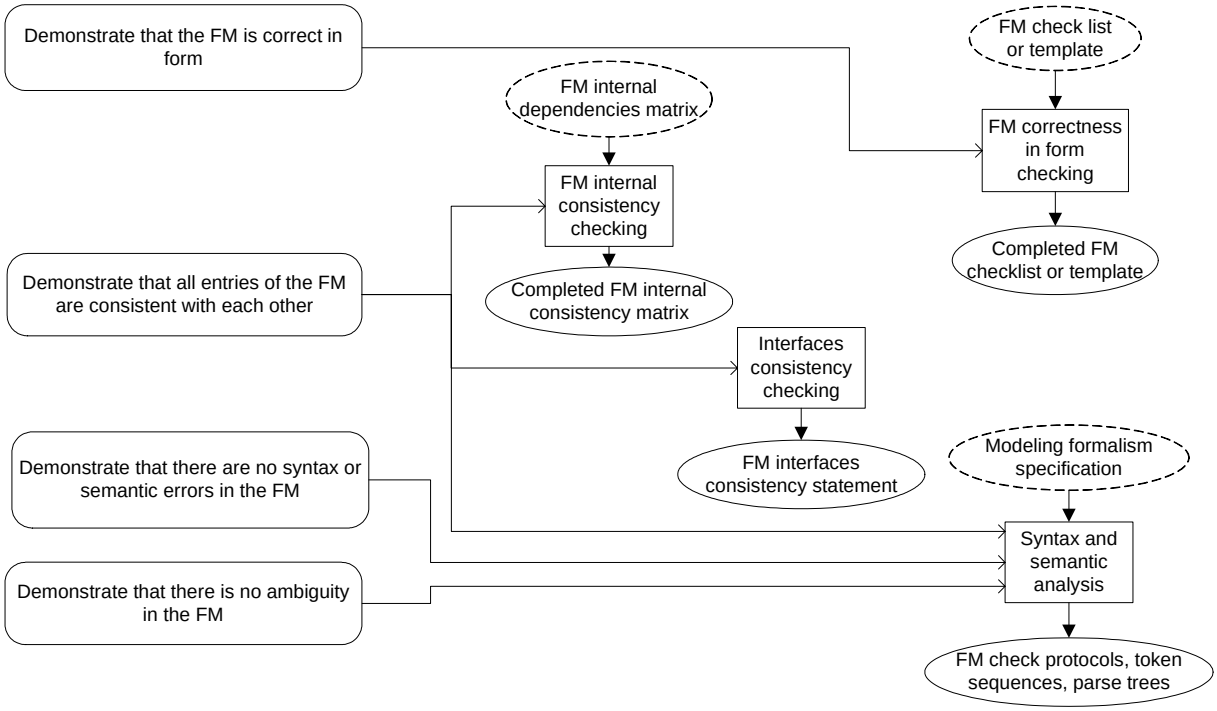


Figure 9: Dependencies between the objectives, techniques, inputs, and outputs of V&V subphase 3.1

- **Proposed techniques for dynamic testing**: N/A
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
  - FM template or checklist
  - Template-based FM internal dependency matrix
  - Modeling formalism specification

- **Provided output**:
  - FM check protocols, token sequences, parse trees
  - Completed FM checklist or template
  - Completed FM internal consistency matrix
  - FM interfaces consistency statement

4.5 CM to FM Transformation V&V

This section documents the goals of V&V sub-phase 3.2, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 10 visualizes their dependencies.

- **Required stage of model development**: Formalization phase completed
- **IP counterpart** (transformation V&V only): CM
- **Sub-phase goal**: Demonstrate completeness of the FM in content and consistency with the CM. This sub-phase aims mainly to ensure that the complete CM was correctly formalized. Explicitly addressed errors include that
  - the FM (including supplemental information and the formally specified experimental framework) is incomplete (i.e. it does not cover all contents of the CM completely), or inconsistent with the CM (i.e., the contents of the CM are not reflected correctly), and
  - the chosen formalism is unsuitable (inefficient)
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this V&V sub-phase. It needs to be ensured that all structural and behavioral contents of the CM are appropriately reflected by the FM, otherwise it cannot be used as a foundation for subsequent efficient implementation.
- **Impact of error detection on model development**: Regress by one phase. Model formalization has to be (at least partially) repeated.
- **Sub-phase objectives**: Demonstrate that
  - the FM completely and consistently reflects the symbolic structure and behavior description of the CM;
  - further (formalism enforced) abstraction and idealization is suitable and correct (algorithms, approximation methods, data);
  - the formally specified experimental framework is complete and consistent with the conceptually described experimental framework of the CM, including input models and distribution functions.
- **Proposed techniques for static analysis**:
  - CM-FM completeness and consistency checking: A trace is created from each content of the CM to the associated content of the FM, and documented in the CM-FM traceability matrix. Contents in the CM without FM counterparts (and vice versa) need to be resolved. A template-based CM-FM dependency matrix can be created and used for this purpose.
  - Assertions insertion: The extracted behavior propositions from V&V spb-phase 2.1 are inserted as annotations or assertions in the appropriate places of the FM. During analytical solution, mental interpretation, or automated interpretation, these assertions are checked for violation. By a formal proof or symbolic execution it is shown that an assertion that serves as pre-condition is correctly transformed into its succeeding post-condition (inductive assertions).
  - Object flow analysis: Possible modifications of an object class by other objects or submodels (and, if possible their sequential order of occurrence) are identified and denoted as an FM object flow graph. This technique is derived from data flow analysis for program code as in [Myers 1979]. The object flow graph

is interpreted and compared to the object behavior described in the CM. This supports the identification of missing or inappropriate specification of modifications of an object or submodel during its life.

- Control flow analysis: By extracting FM control flow graphs, the internal behavior alternatives within the formal submodels are made explicitly visible [Carr and Balci 2000]. They are compared to the submodel behavior options described in the CM. Incorrect formal specification of decision making within the formal submodels can be detected this way.
- Model checking: The CM behavior propositions from V&V sub-phases 2.1 and 2.2 are formalized using an appropriate temporal logic, and transferred into acceptance automata. Subsequently it is proven that the complete language generated by the associated formally specified (state transition) submodel is accepted by the corresponding acceptance automaton [Katoen 1999] (see also main document, section 5.3).

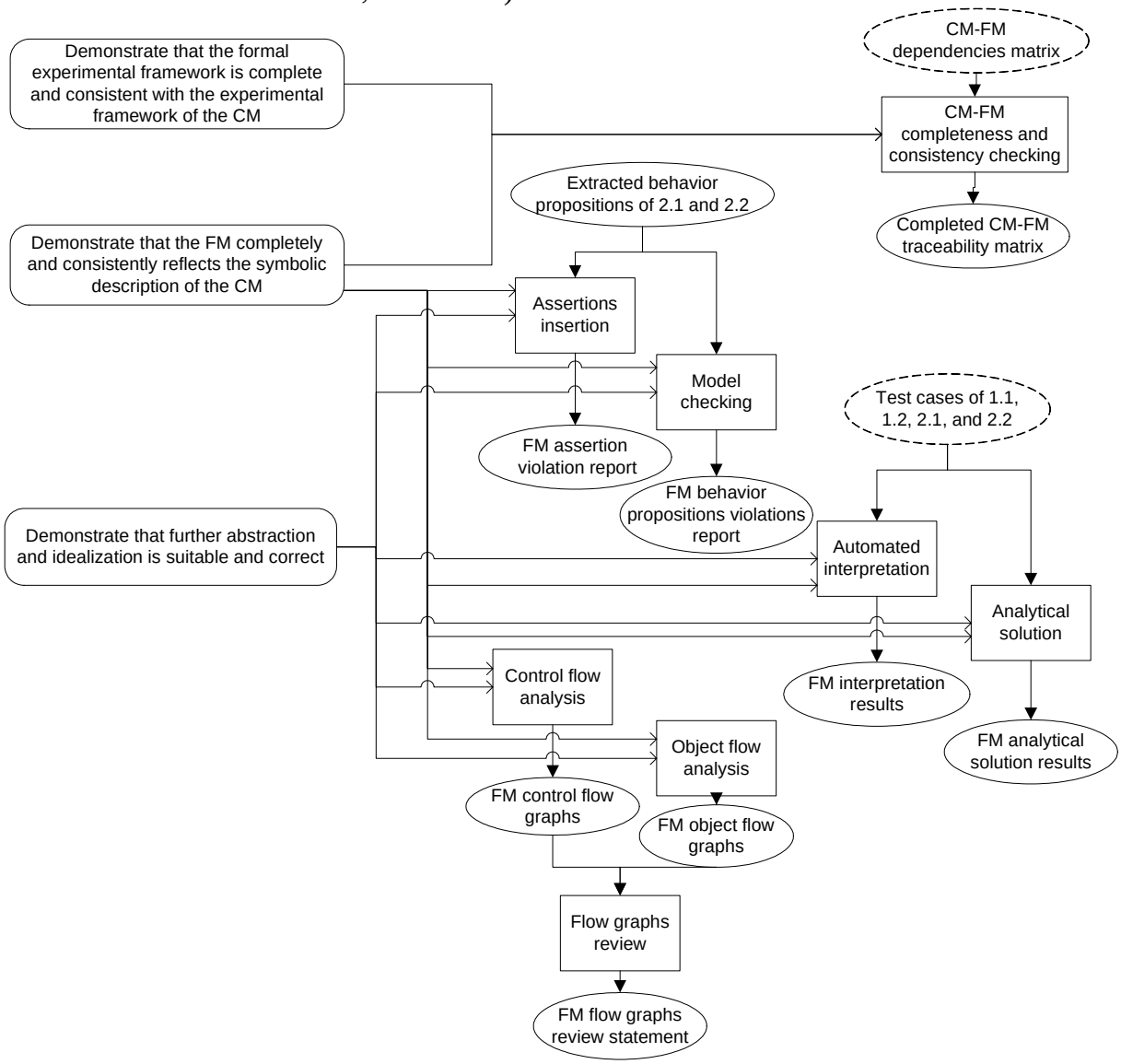


Figure 10: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 3.2

- **Proposed techniques for dynamic testing**:

- Testing by automated interpretation: Assuming the existence of a suitable execution environment, a subset of the test cases defined in V&V sub-phases 1.1, 1.2, 2.1, and 2.2 can be executed by interpreting the FM. Depending on the performance of the FM execution, testing techniques used for the EM can be applied to the FM. In this case, the FM plays the role of a "prototype" of the EM.
- Analytical solution: This is no true testing, although model output (results) is created. If the modeling formalism supports the analytical solution for at least a subset of the input combinations defined in the test cases of V&V sub-phases 1.1, 1.2, 2.1, and 2.2, the results created this way can be used for assessment of consistency with the anticipated behavior of the CM.

- **Proposed techniques for results evaluation**:
  With small amounts of computer generated simulation results or analytically gained results available, evaluation methods applied to the EM can be used here, too. It is most likely that statistical methods cannot (yet) be applied.
- **External references, leveraged information, and reused V&V output**:
  - CM-FM template-based dependency matrix
  - Extracted behavior propositions of 2.1 and 2.2
  - Test cases of 1.1, 1.2, 2.1, and 2.2
- **Provided output**:
  - Completed CM-FM traceability matrix
  - FM object flow graphs and comparison statements
  - FM control flow graphs and comparison statements
  - FM assertions violations report
  - FM behavior propositions violations report
  - FM interpretation results
  - FM analytical solution results
  - FM flow graphs review statement

4.6 SPD to FM Transformation V&V

This section documents the goals of V&V sub-phase 3.3, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 11 visualizes their dependencies.

- **Required stage of model development**: Formalization phase completed
- **IP counterpart** (transformation V&V only): SPD
- **Sub-phase goal**: Confirm completeness of the FM in content and consistency with the SPD. This sub-phase mainly aims to confirm that the model behavior meets the requirements, using the limited amount of quantitative model output now available. Addressed errors also include
  - violation of explicit formalization requirements and
  - incompliance of the FM to required higher level formal framework
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. It can be skipped, if it is credible that all contents of the SPD are covered by the CM (V&V sub-phase 2.2), and that the FM completely and consistently reflects the CM (V&V sub-phase 3.2). However, in addition to the activities of V&V sub-phase 2.2, a more rigorous assessment concerning the model's satisfaction of required behavior can be made, if quantitative data for a subset of input combinations is available.
- **Impact of error detection on model development**: Regress by two phases. If this sub-phase reveals that a required content is missing or wrong in the FM, an error in ei-

ther the system analysis or the model formalization is detected. Both phases have to be repeated (at least partially).

- **Sub-phase objectives**: Confirm that
  - the FM is complete in content and consistent with the SPD;
  - the symbolic behavior specification in the FM is as accurate as required; and
  - the FM is compliant with the required formal framework.
- **Proposed techniques for static analysis**:
  - SPD-FM completeness and consistency checking: By reusing the SPD-CM and CM-FM traceability matrices from V&V sub-phases 2.2 and 3.2, and using a template-based SPD-FM dependency matrix, a trace from each requirement to the associated FM contents is created, and documented in the SPD-FM traceability matrix. Requirements without associated FM contents (and vice versa) indicate a problem.
  - Object flow graph and control flow graph review: The flow graphs of V&V sub-phase 3.2 are reviewed for consistency with the structural and behavioral requirements stated in the SPD. Incompliance is documented in the flow graph compliance statement.

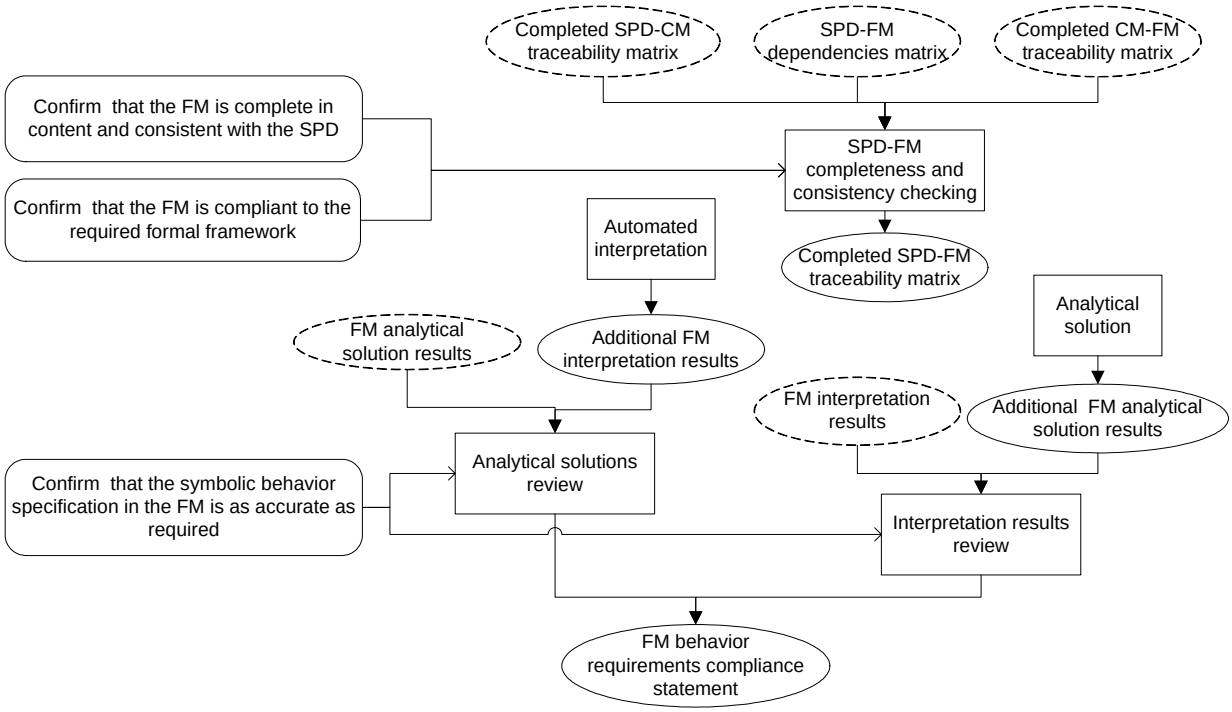


Figure 11: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 3.3

- **Proposed techniques for dynamic testing**:
  - Automated interpretation: Generation of additional FM behavior data for additional test cases, if required.
  - Analytical solution: Computation of additional FM behavior data for additional test cases, if required.
- **Proposed techniques for results evaluation**:
  - Analytical solutions review: Experts analyze, whether the analytical solution of the model indicates model behavior as required, if possible by comparison to "hard" quantitative test criteria. The review result is documented as a behavior accuracy statement.

    - Interpretation results review: Experts analyze, whether the interpretation of the model indicates model behavior as required. The review result is documented as a behavior accuracy statement.
- **External references, leveraged information, and reused V&V output**:
    - FM analytical solution results from 3.2
    - FM interpretation results from 3.2
    - CM-FM traceability matrix from 3.2
    - SPD-CM traceability matrix from 2.2
    - Template-based SPD-FM dependency matrix
- **Provided Output**:
    - Completed SPD-FM traceability matrix
    - Additional FM interpretation results
    - Additional FM analytical solution results
    - FM behavior requirements compliance statement

4.7 SN to FM Transformation V&V

This section documents the goals of V&V sub-phase 3.4, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 12 visualizes their dependencies.

- **Required stage of model development**: Formalization phase completed
- **IP counterpart** (transformation V&V only): SN
- **Sub-phase goal**: Confirm completeness of the FM in content and consistency with the SN. This sub-phase mainly aims to confirm that the FM is the appropriate step towards the satisfaction of the SN, using the limited amount of available quantitative model results.
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. It can be skipped, if it was assured previously that the SPD covers the SN (1.2), and the FM meets the SPD (3.3). However, if the FM allows interpretation and visualization of the model behavior, in this stage direct feedback with the sponsor is extremely valuable. (This is related to the concept of "rapid prototyping", where prior to developing the deliverable product an executable prototype is presented to the sponsor as soon as possible.)
- **Impact of error detection on model development**: Regress by three phases. The FM misses an important aspect of the SN. Thus, the (at least partial) repetition of the problem definition, system analysis and model formalization is required.
- **Sub-phase objectives**: Confirm that
    - the model border allows the needed experimentation (stimulation and observation), and
    - the FM behavior description (including embedded data) is as accurate as needed.

  Remark: The objectives are similar to those of V&V sub-phase 3.3, with the only difference not assessing the compliance with the SPD, but the adequacy with respect to the real needs.
- **Proposed techniques for static analysis**:
    - SN-FM completeness and consistency checking: By reusing the SN-SPD and SPD-FM consistency matrices from V&V sub-phases 1.2 and 3.3, and the use of a template based SN-FM dependency matrix, a trace from each need to the associated FM contents is created, and documented in the SN-FM consistency matrix. Needs without associated FM contents (and vice versa) indicate a problem.

- Object flow graph and control flow graph review: The flow graphs of V&V sub-phase 3.2 are reviewed for consistency with the SN. Inadequacy is documented in the flow graph adequacy statement.

- **Proposed techniques for dynamic testing**:
  - Interpretation or analytic solution of ad hoc test cases: With the possibility of first (limited) experimentation with the model, the sponsor may describe meaningful new test cases, which previously have not been identified, yielding additional results.
- **Proposed techniques for results evaluation**:
  - Visualization and Face Validation: If quantitative model results are available from analytic solution or interpretation (e.g., taken from V&V sub-phase 3.2), they are visualized and evaluated for plausibility by experts, which are involved in the solution of the real problem the model or the simulation results are supposed to contribute to.

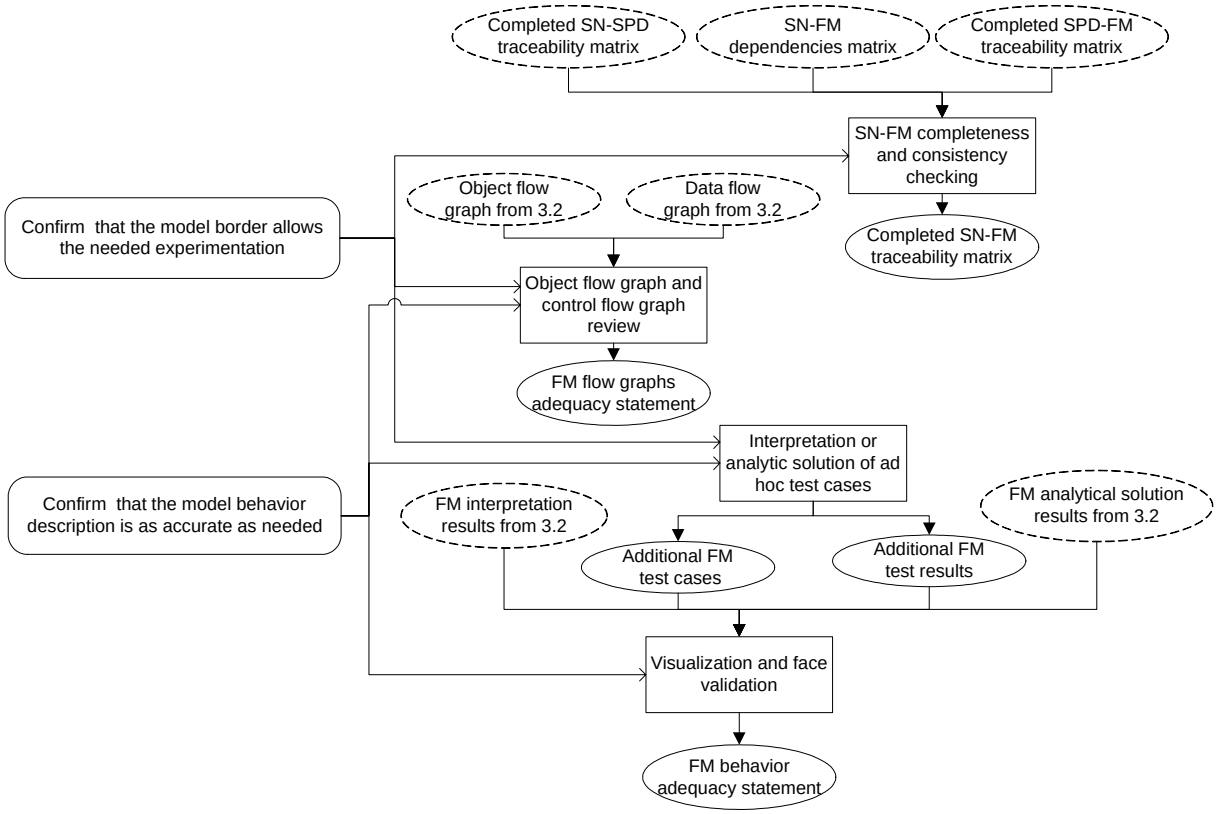


Figure 12: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 3.4

- **External references, leveraged information, and reused V&V output**:
  - Completed SN-SPD traceability matrix
  - Completed SPD-FM traceability matrix
  - Template-based SN-FM dependency matrix
  - Object flow graph from 3.2
  - Data flow graph from 3.2
  - FM interpretation results from 3.2
  - FM analytical solution results from 3.2
- **Provided output**:
  - Completed SN-FM traceability matrix
  - FM flow graph adequacy statement

- Additional FM test cases
- Additional FM results
- FM behavior adequacy statement

5 V&V OF THE EXECUTABLE MODEL

The EM consists of the executable symbolic description of the model (i.e., a specification that automatically can be compiled or interpreted by a computer, e.g., program code), and a platform and environment for executing the model, which allows the generation of SR. Whenever the EM is executed during EM V&V, this is exclusively done for the purpose of detection of incorrectness and unsuitability. This must not be confused with the execution of the model during the experimentation phase, where – under the assumption that the EM is correct and suitable – SR are generated to trigger conclusions for the real world.

5.1 Expected Implementation Activities

When preparing an FM for computer-based simulation, a simulation method must be selected in accordance with the state space description and representation of time chosen in the CM, and the solution methods suitable for the FM (continuous vs. discrete simulation, see also main document, section 1.2.4). During the implementation phase, a simulation environment or infrastructure is used or created, and the FM is transformed into a specification that can be interpreted or executed within the simulation environment.

Implementation failures occur, when transforming the FM into a computer executable specification. Implementation errors summarize all “technical” errors, including software/hardware design errors, coding errors, compiler errors, installation errors, interoperability problems, non-compliance with interface or network communication standards, and runtime errors. (Most of the below error types are also characteristic for software applications that do not implement simulation models).

The EM is created by the programmer and will be run by the user. When documenting the EM, the population of the associated sub-spaces in the three-dimensional model information space introduced in the main document, section 4.3.1 is desirable. There must not be any deviation from the FM in this phase. Neither qualitative nor quantitative information should be added to the model, only technical aspects, i.e., calls of operating systems procedures for I/O from files, memory allocation, or the selection of data type for attributes are addressed. Input data needs to be made available electronically.

- **Implementation to specification failure**: The FM is not transformed correctly and completely into an executable version.
- **Software engineering failure**: The software development method is error-prone, programming conventions are violated, software quality assurance measures are insufficient, the programming language is antiquated, or sound software engineering methods are ignored.

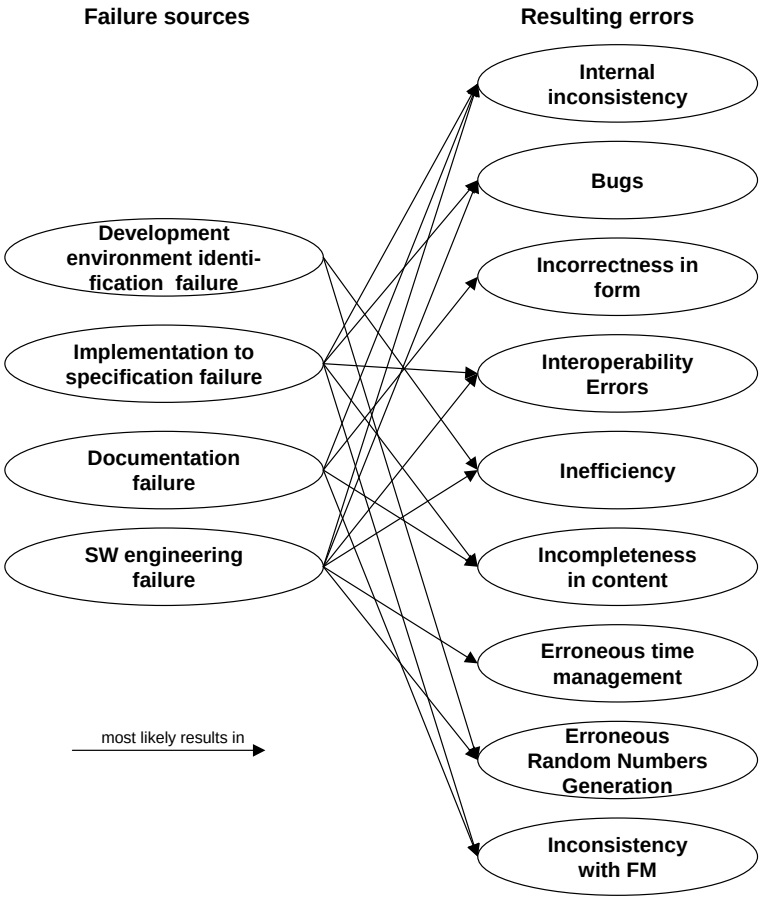


Figure 13: Dependencies between failures in the implementation phase and errors in the EM

- **Development environment selection failure**: Modules or functional components frequently used for simulation are provided in form of *simulation languages*, *simulation libraries*, or *simulation environments*. Simulation languages (e.g., Simula [Lamprecht 1976], SLX [Henriksen 1998]) feature specialized language elements or constructs that directly support encoding of simulation models, while simulation libraries (e.g., DEVSJava [DEVSJAVA 2.7 2003], GPSS Fortran [Schmidt 1985]) provide previously encoded, simulation specific data structures, objects, and functions for high level programming languages. Simulation environments (e.g., Visual Simulation Environment, SLX, Simul8) usually support development and execution of models by providing a simulation language (or predefined simulation specific elements), and, e.g., component catalogues, drag-and-drop modeling, a control panel for running and debugging the simulation, and results analysis functions. This support may be domain or problem class oriented, with highly specialized predefined elements. It also may be less constraint for multiple applications by definition of more general elements, or as flexible as any higher programming languages by simply augmenting one with the desired simulation functionality. The quality of the used "standard" components heavily influences the quality of the EM, but their broader application has a positive impact on their perceived correctness.
- **Documentation failure**: The importance of the process of EM documentation is underestimated, and not all relevant information about the EM is recorded.

The implementation phase is re-entered, when the FM changes or an error in the EM is detected.

5.2 EM Documentation Requirements

The compiled/interpretable program specification in its operating environment finally makes up the EM that receives input data and creates simulation output or SR. Required contents are (a sample EM is given in Appendix C):

1) **Configuration control information**: Required for configuration control, references to preceding IPs, and references by succeeding IPs. It should include an unique version identifier.
2) **Runtime environment**: Description of the platform, operating system, additionally required running processes or applications (e.g., communication infrastructure), or recording tools and data bases.
3) **Software description**:
   a) **Identification of supporting software**: Simulation language, simulation libraries, or simulation environment.
   b) **High level design**: This is an extension of the FM, with the high level design of simulation control added. Distribution information for distributed execution is included here.
   c) **Low level design**: This is the refinement of the high level design as a direct preparation for the implementation.
   d) **Commented executable specification**: Program code or other specification that is executed or interpreted.
4) **Simulation infrastructure**: This includes all added simulation functionality as
   a) **Simulation control and time management**: The dynamic change of the state parameters over time and the advancement of simulation time need to be managed. Structure and functionality of simulation control depend on the simulation method.
   b) **Random number generator**: For stochastic simulation random numbers are required, which are created by the RNG.
5) **Structure and behavior description**: Implementation of all submodels on each layer, their internal behavior, their decomposition into lower-level submodels, and their interactions. This includes *for each submodel on each layer*:
   a) **Formal Submodel**: Identification of the associated formal submodel.
   b) **Origin**: Identifies the executable submodel as custom-made or as an imported executable submodel (e.g., a reused federate in a federation). If a pre-defined submodel is used, a rationale for integrating it is required, e.g. identification of an interoperability standard, which demands its use, or assumed advantage of using it.
   c) **Interface**: This includes
      i) the description of I/O streams, file formats, callback functions, exchangeable objects, remote function calls, or communication library function calls, parameter types, update conditions, accuracy of parameters, and the assignment to the associated entry in the FM;
      ii) justification for the chosen formats and types.
   d) **Structure and internal behavior**: This includes
      i) the identification of all integrated executable submodels;
      ii) state variable data types, update conditions, accuracy;
      iii) inheritance hierarchy of objects within the submodel;
      iv) update procedures for state variables and output parameters;
      v) conditions for or frequency of calls of update procedures;
      vi) assignments to associated entries in the FM.
   e) **Interaction with other submodels**: This includes the implementation of internal dynamics of subsystems and objects, state variables, state transitions, where applicable.
   f) **Electronic data sources**: Identification of accessible data bases, access protocols and data conversion requirements.

6) **Standards summary**: Identification of any standard the EM is compliant with.

There should be a justification for each number no information is given for. The required information may be completed during several iterations of the implementation phase.

5.3 Potential Errors in the EM

The dependencies between the identified failures during the implementation and errors in the EM are sketched in Figure 13. Potential errors identified for the EM are introduced in the following:

- **Incorrectness in form** (internal error): An EM should contain – besides the program code – the high- and low-level design of the code, including module descriptions, (program) objects hierarchies, state transition diagrams, and a precise I/O interface specification. For the assessment of formal correctness and completeness, there must be a clearly identified reference form, summarizing the mandatory contents of the EM, as e.g. provided in section 5.2.
- **Internal inconsistency** (internal error): The contents of the EM are not consistent, e.g., the code does not reflect the high level design, the code is not executable in the simulation environment, or function or object libraries are not linkable.
- **Technical interoperability error** (internal error): The EM is not able to communicate with other components as intended. The EM is not compliant with desired standards, e.g., uses the wrong Object Request Broker version, or does not satisfy HLA compliance requirements [IEEE 1516.1 2001].
- **Bugs** (internal error): To compile and execute (or interpret) the program code, it must be syntactically and semantically correct. The reference for this check is the unambiguous specification of the programming or simulation language. Even, if syntactical and semantic correctness is ensured, errors may be hidden in the program code that result in absurd simulation results or program crash. The reason for this is not necessarily a programming error – an instable operating system, parallel processes, unexpected malfunctioning of periphery devices may cause runtime errors [Echtle 1990].
- **Time management errors** (internal error): The advancement of simulation time is faulty. For example, for a discrete event simulation, wrong insertion of events into the events list (especially for distributed simulation), or wrong handling of the event list. For a time-step simulation, attributes that change their values simultaneously with respect to simulation time are updated sequentially, causing undesired temporal dependencies. The simulation method does not allow to observe the desired properties. The discrete event simulation may not be scaled to wall clock time (where applicable), or the “samples” of the states created by a discrete simulation are not dense enough.
- **Correlated random numbers** (internal error): The random number generator creates correlated random numbers, causing undesired dependencies in the simulation output. [L’Ecuyer 1998] for example discusses several methods how to generate random numbers. Again, this functionality is easily encapsulated in a module, which needs to be attached to the EM during the implementation phase. Being capable of examining the random number generator for randomness is essential for V&V of the EM.
- **Inefficiency** (internal error): The EM may simply be to slow to produce simulation results in time (*Violation of real time requirements*).
- **Unsuitable simulation method** (transformation error): Simulation methods can be distinguished by the type of sampling or discretization of time that they support (time-step simulation vs. discrete event simulation, see also section 1.3.2), and their *strategy for time advancement*. Especially in a distributed simulation, several strategies for time advancement are available, including *conservative and optimistic strategies* [Fu-

jimoto 2000]. A simulation control unit is often made available as a reusable component. To assess the correctness and suitability of an EM, insight into the time advancement mechanism and its implementation is required, which also should be rigorously tested.

- **Incompleteness in content** (transformation error): The EM does not implement the FM correctly and completely.
- **Inconsistency with the FM** (transformation error): The contents of the EM are not consistent with their associated contents of the FM.

The absence of the above errors shall be demonstrated by the implementation of the below V&V activities.

5.4 EM Internal V&V

This section documents the goals of V&V sub-phase 4.1, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 14 visualizes their dependencies.

- **Required stage of model development**: Implementation phase completed
- **IP counterpart** (transformation V&V only): N/A
- **Sub-phase goal**: Demonstrate correctness in form and self-consistency of the EM. This includes the demonstration that the EM is "good" software and that the implemented simulation infrastructure (including time management and random number generation) was built to its specification. Addressed Errors include
    - software "bugs", including syntactical and semantic coding errors, runtime errors, and concurrency errors;
    - technical interoperability problems;
    - unsuitable use of random numbers; and
    - simulation infrastructure errors, including time management errors and errors in random number generation.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. Experimentation and further V&V activities rely on a "properly running" EM, and software errors appearing later in the development process will delay and distort the observation of model behavior.
- **Impact of error detection on model development**: Regress by one phase. Implementation must be (at least partially) repeated.
- **Sub-phase objectives**: Demonstrate that
    - the EM, its supplemental information, and the implemented experimental framework are correct in form (i.e., consistent with the EM documentation requirements),
    - the contents of the EM and its supplemental information are consistent with each other,
    - the EM is stable, reliable, and testable software (i.e., consistent with Software Engineering "best practice"),
    - the simulation infrastructure (including time management and random number generation) is consistent with its specification, and
    - the distributed executable submodels are technically interoperable, if applicable.

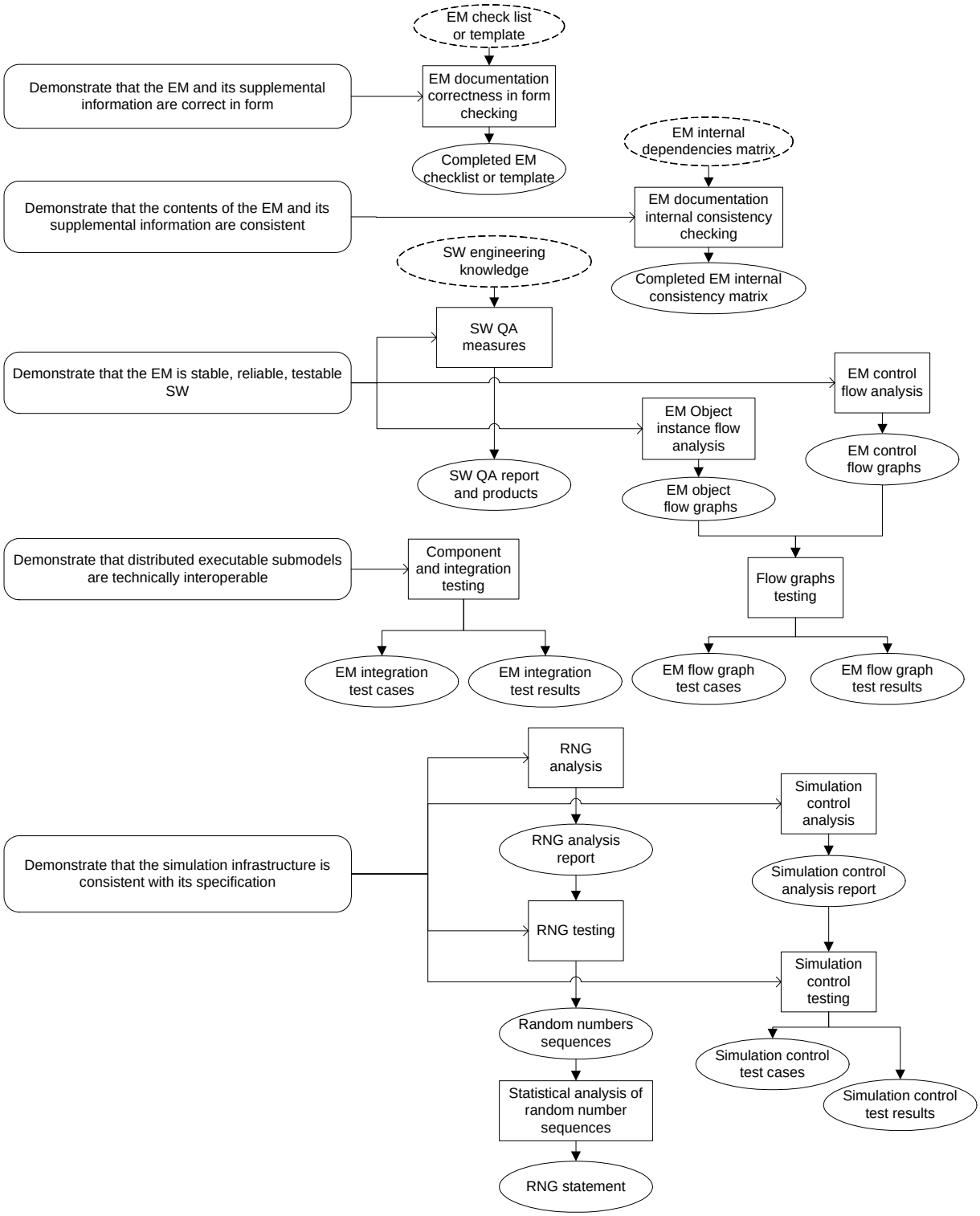


Figure 14: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 4.1

- **Proposed techniques for static analysis**:
  - EM documentation correctness in form checking (includes completeness checking): Using a checklist or a template as provided in 5.2, the EM and its supplementary information are checked for completeness and consistency with the template. This yields a completed EM checklist or template.

- EM documentation internal consistency checking: Using a of a template-based EM internal dependency matrix, the EM documentation is checked for internal consistency. This results in a completed EM internal consistency matrix.
- Software QA measures: To ensure sufficiently high quality of the software, quality increasing measures taught in software engineering should be taken. These techniques, including debugging, desk checking, proof of correctness, and many others are discussed in literature [Liggesmeyer, Sneed, and Spillner 1992; Myers 1979; Beizer 1990].
- EM control flow analysis: Using appropriate tools, the control flows of the EM (or at least the executable submodels) are extracted and documented as EM control flow graphs. Classically, these control flow graphs are used for the creation of high-yield test cases for software QA, but will also be reused during subsequent V&V activities.
- EM object instance flow analysis: The potential flows of atomic executable submodels are extracted from the code, documented in EM object flow graphs, and appropriate test cases are created to allow the observation of the object along these flow paths. Besides for the detection of pure software errors, these object flow graphs can later be reused for object flow comparison.
- RNG Analysis: The code of the random number generator is reviewed. From the structure of the RNG statements about its properties can be derived [L'Ecuyer 1998], e.g. an upper limit for its period length can be determined.
- Simulation control and time management analysis: As one of the core pieces of the EM, the simulation control unit is individually and rigorously analyzed.

- **Proposed techniques for dynamic testing**:
  - Software test techniques: Software Engineering offers numerous methods for software testing, including module testing, integration testing, special input testing, and others. These methods are used to increase the availability and reliability of software and are documented in the appropriate literature, as, e.g., [Myers 1979] or [Beizer 1990].
  - EM control flow and EM object flow testing: Based on the EM control flow graphs and the EM object flow graphs test cases are created and executed for the purpose of software error detection.
  - RNG testing: The RNG is ran, and its behavior is recorded as sequences of random numbers.
  - Simulation control and time management testing: The simulation control unit is individually tested, whether it is able to control the dynamic change of state of the EM as specified. Special care is required by the resolution of concurrent events or concurrent changes of state in diverse submodels.
- **Proposed techniques for results evaluation**:
  - Statistical analysis of random number sequences: Literature [L'Ecuyer 1998; Hellekalek 1997; Keppler 1993] provides various techniques combinations for RNG results evaluation.
- **External references, leveraged information, and reused V&V output**:
  - EM checklist or template
  - Template-based EM internal dependency matrix
  - Software engineering domain knowledge
- **Provided output**:
  - Completed EM checklist or template
  - Completed EM internal consistency matrix
  - Software QA report and products
  - EM control flow graph

- EM object flow graph
- EM flow graph test cases and test results
- RNG analysis report, random numbers sequences, and RNG statement
- EM integration test cases and EM integration test results
- Simulation control analysis report
- Simulation control test cases
- Simulation control test results

5.5 FM to EM Transformation V&V

This section documents the goals of V&V sub-phase 4.2, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 15 visualizes their dependencies.

- **Required stage of model development**: Implementation phase completed
- **IP counterpart** (transformation V&V only): FM
- **Sub-phase goal**: Demonstrate completeness of the EM in content and consistency with the FM. This sub-phase aims to ensure that the implementation of the simulation infrastructure is suitable for the underlying modeling formalism, and that the EM implements the FM completely and consistently. Addressed errors include:
  - the implementation of the simulation infrastructure is inconsistent with the used modeling formalism;
  - the submodel structure, the internal behavior, and interactions are incomplete or inconsistent with the FM;
  - distribution functions specified in the FM or the formally specified experimental framework are not matched by the observed associated empirical distributions of the EM.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. If it is not credible that the software implements the formally specified model, it must be doubted that the modeling theory underlying the used formalism was adequately applied for model solution.
- **Impact of error detection on model development**: Regress by one phase. Implementation must be (at least partially) repeated.
- **Sub-phase objectives**: Demonstrate that
  - the chosen implementation technique of the simulation infrastructure is suitable for the formalism the FM is based on,
  - the EM structure is complete and consistent with FM,
  - the EM implements the internal behavior and interactions completely and consistently with the FM,
  - all random number distributions are implemented as defined in the FM, and
  - data types and algorithms are transformed to digitally computable form acceptably
- **Proposed techniques for static analysis**:
  - FM-EM completeness and consistency checking: A trace is created from each content of the FM to the associated content of the EM, and documented in an FM-EM traceability matrix. Contents in the FM without EM counterparts (and vice versa) need to be resolved. A template-based FM-EM dependency matrix can be created and used for this purpose.
  - Flow graph comparison: The FM flow graphs from V&V sub-phase 3.2 are compared to their EM counterparts from V&V sub-phase 4.1. Differences indicate a problem and need to be resolved.

- Simulation infrastructure analysis: It is examined, whether simulation control is implemented according to one of the supported solution methods of the modeling formalism.
- Interface analysis: By analyzing the interfaces of the executable submodels it is determined, whether their communication takes place according to the communication means provided by the formalism.
- Distribution generators analysis: RNG create uniform distributions, which are transformed into the desired distribution by a mathematical function. By review of the RNG code, it is analyzed, whether the underlying mathematical function is erroneous.

- **Proposed techniques for dynamic testing**:
  - Analytical solution duplication: According to the test cases subsets chosen in 3.3 and 3.4, the EM is run under the same input conditions under which the FM was analytically solved.
  - FM interpretation duplication: According to the test cases subsets chosen in 3.3 and 3.4, the EM is run under the same input conditions under which the FM was executed by interpretation.
  - Goodness of fit testing: The empirical distribution functions defined by "random" numbers observed during testing are recorded for subsequent comparison to their associated distribution functions in the FM.
  - Assertion checking: All behavior propositions specified within the FM are embedded into their associated positions in the EM. Their violation is automatically detected during the execution of tests.
- **Proposed techniques for results evaluation**:
  - FM-EM results comparison: The model results generated by analytical solution or interpretation of the FM are compared to the associated results of the EM. This can be achieved by, e.g., graphical visualization or statistical comparison.
  - Statistical tests: These are used to determine to, which degree the empirical distribution functions created from the recorded number sequences fit their associated theoretical distribution functions, or another empirical distribution function.
- **External references, leveraged information, and reused V&V output**:
  - Simulation method
  - FM interpretation test cases and results from 3.3 and 3.4
  - Analytical solution test cases and results from 3.3 and 3.4
  - FM flow graphs from 3.2
  - EM flow graphs from 4.1
  - FM-EM dependency matrix
- **Provided output**:
  - Simulation infrastructure analysis statement
  - Simulation infrastructure test cases
  - Simulation infrastructure test results
  - EM results and I/S/O traces
  - EM-FM results comparison statement
  - Completed FM-EM consistency matrix
  - Flow graph comparison statement
  - Distribution generator analysis statement
  - Goodness of fit statement
  - Interface analysis statement

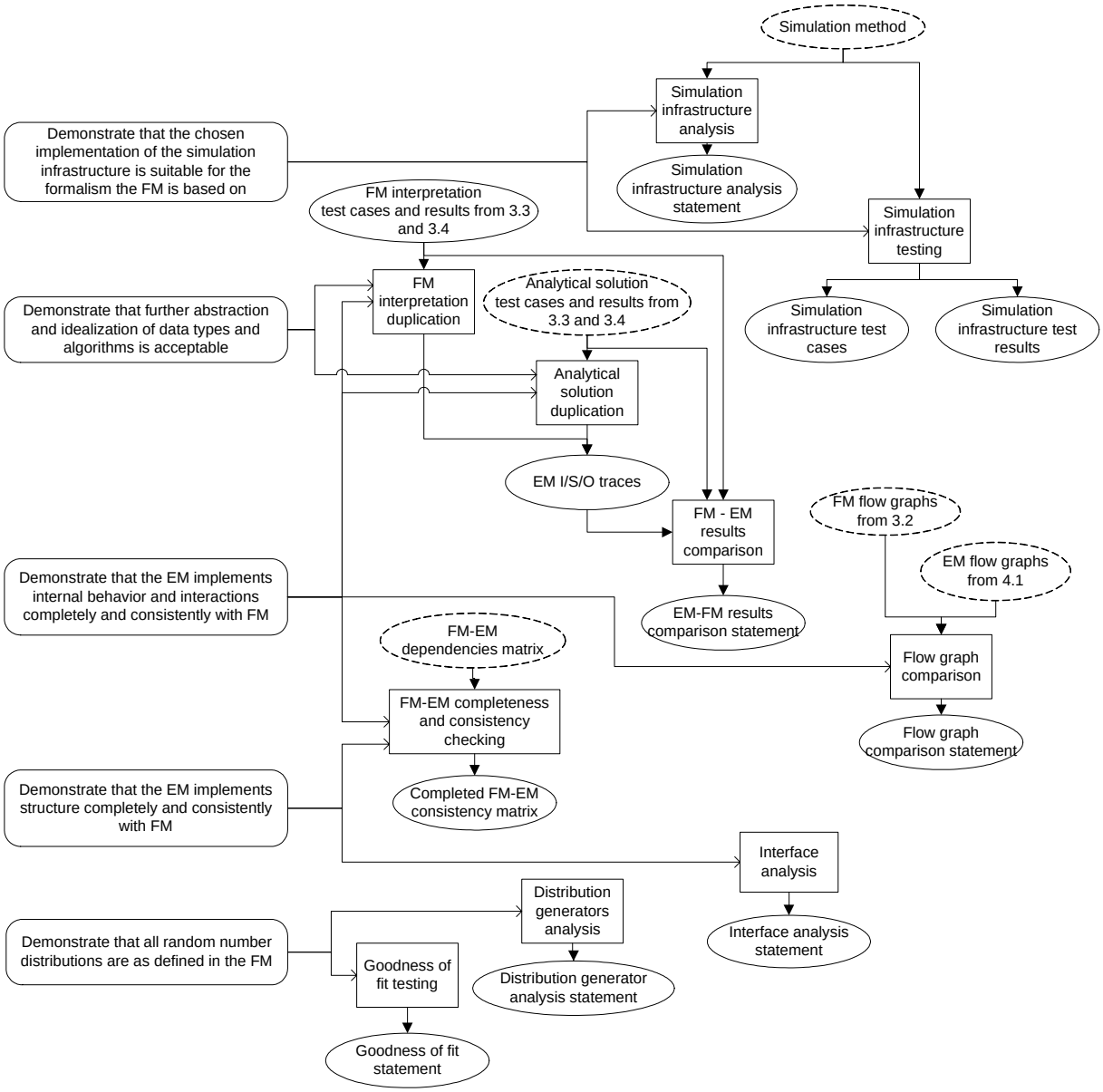


Figure 15: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 4.2

5.6 CM to EM Transformation V&V

This section documents the goals of V&V sub-phase 4.3, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 16 visualizes their dependencies.

- **Required stage of model development**: Implementation phase completed
- **IP counterpart** (transformation V&V only): CM
- **Sub-phase goal**: Confirm completeness of EM in content and consistency with the CM. This sub-phase is mainly concerned with the question, whether the EM actually is an implementation of the modeler's idealized and abstracted perception of the real system. The error mainly addressed is that the EM or its submodels do not show the behavior as intended with the CM.
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. It can be skipped, if it was assured previously that the FM specifies the CM correctly and suitably (V&V sub-phase 3.2), and the EM meets the specification given

by the FM (V&V sub-phase 4.2). However, this is the first opportunity for the user to actually observe model behavior for all desired input combinations, and a valuable (although late) opportunity to detect behavior specification errors or implementation errors.

- **Impact of error detection on model development**: Regress by two phases. Model formalization and model implementation must be (at least partially) repeated.
- **Sub-phase objectives**: Confirm that
  - the executable submodel structure is consistent with the description of the submodel structure provided in the CM;
  - the heuristics and algorithms implemented in the EM for internal submodel behavior and submodel interaction, including their further abstraction for implementation due to digitalization, yield behavior data with sufficient accuracy;
  - the (actually observed) EM behavior is consistent with domain knowledge.
- **Proposed techniques for static analysis**:
  - CM-EM completeness and consistency checking: By reusing the CM-FM and FM-EM traceability matrices from V&V sub-phases 3.2 and 4.2, and by using a template based CM-EM dependency matrix, a trace from each CM content to the associated EM contents is created, and documented in the CM-EM traceability matrix. CM contents without associated EM contents (and vice versa) indicate a problem.
  - Flow graph analysis: The EM control flow graphs and the EM object flow graphs are reviewed and judged for consistency with the CM and additional domain knowledge.
- **Proposed techniques for dynamic testing**:
  - Reference experiments execution: Run test experiments for the EM or its executable submodels of which the modeler has clear expectations about their outcome (oracle), as defined in 2.1. EM behavior is recorded as EM I/S/O traces (see main document, section 5.3.8).
  - Assertion checking: The behavior propositions of V&V sub-phase 2.1 are inserted into the EM and all of its submodels. During execution, violations are automatically reported.
  - Sensitivity analysis: Several experiments with the EM are conducted under slightly modified input conditions. Thereby the sensitivity of the EM input parameters is determined, and subsequently compared to the perceived relevance of the real influences to the real system (system knowledge) [Kleijnen 1998].
  - Fault-Failure-Insertion: The EM is modified according to the CM modifications from 2.2 to produce erroneous model behavior. If the model does not behave as predicted, this problem needs to be resolved. EM behavior is recorded as an EM I/S/O trace.
- **Proposed techniques for results evaluation**:
  - I/S/O trace analysis: Traces of the EM behavior are analyzed for violations of behavior propositions specified in 2.1. This can be automated to some extend (see also main document, section 5.3.8). The assessment of event sequences that are generated by the EM, and their correspondence to real system behavior is also a possible I/S/O trace analysis activity.
  - Visualization and face validation of EM I/S/O traces: The behavior of the EM and its executable submodels is visualized in a form that can be intuitively understood and judged by SMEs. A possible form of results evaluation is the Turing test, where the SMEs compare I/S/O traces measured at the real system and EM I/S/O traces without knowing the sources of the traces. During the Turing

test, they categorize the traces as EM-originated or real-system-originated, and explain the reasoning for the assignment.

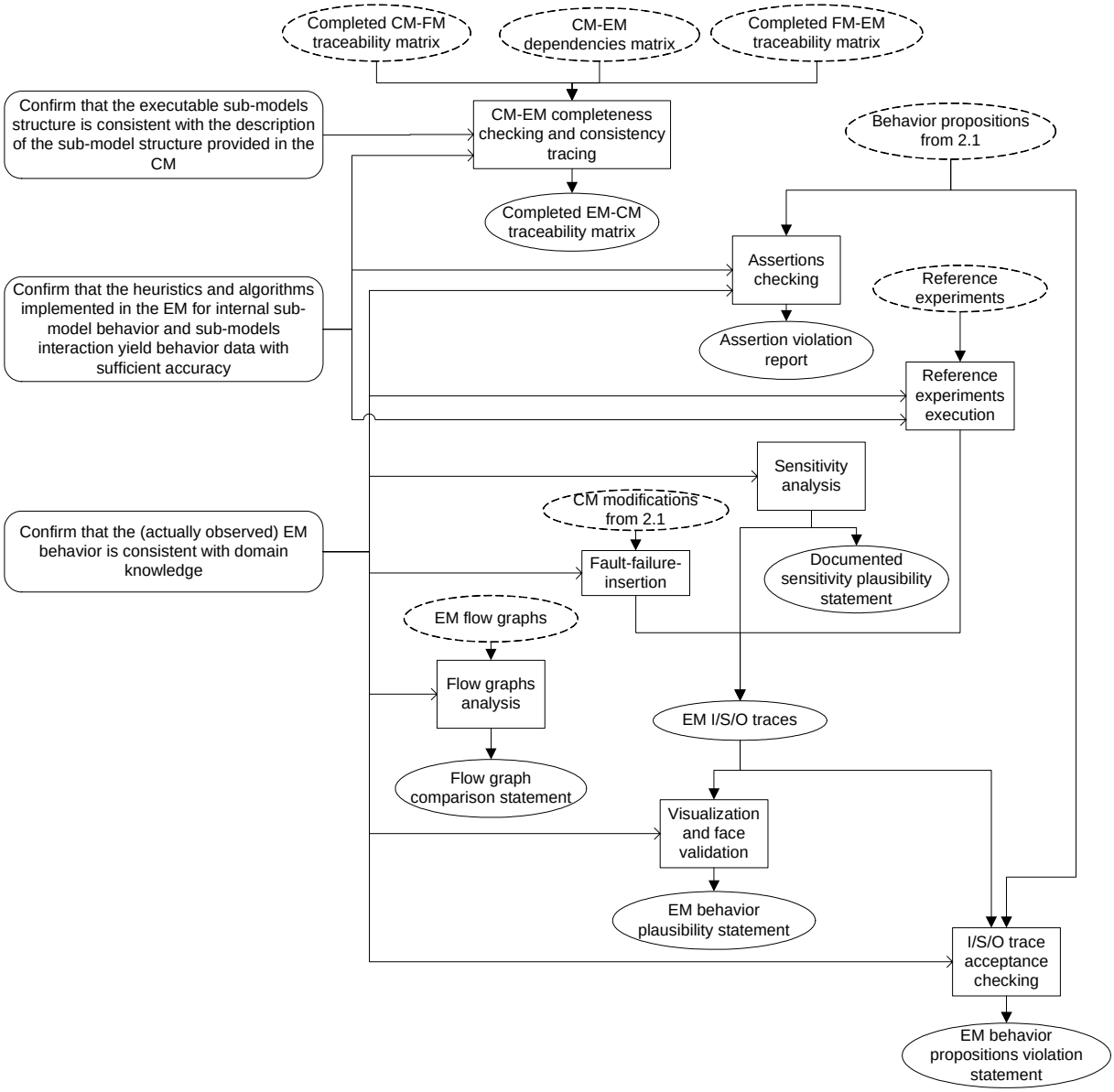


Figure 16: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 4.3

- **External references, leveraged information, and reused V&V output**:
  - Completed CM-FM traceability matrix
  - CM-EM dependency matrix
  - Completed FM-EM traceability matrix
  - Behavior propositions from 2.1
  - Reference experiments
  - CM modifications from 2.1
  - EM flow graphs
- **Provided output**:
  - Completed EM-CM traceability matrix
  - Assertion violations report
  - Documented sensitivity plausibility statement

- Flow graph comparison statement
- EM I/S/O traces
- EM behavior plausibility statement
- EM behavior propositions violation statement

5.7 SPD to EM Transformation V&V

This section documents the goals of V&V sub-phase 4.4, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 17 visualizes their dependencies.

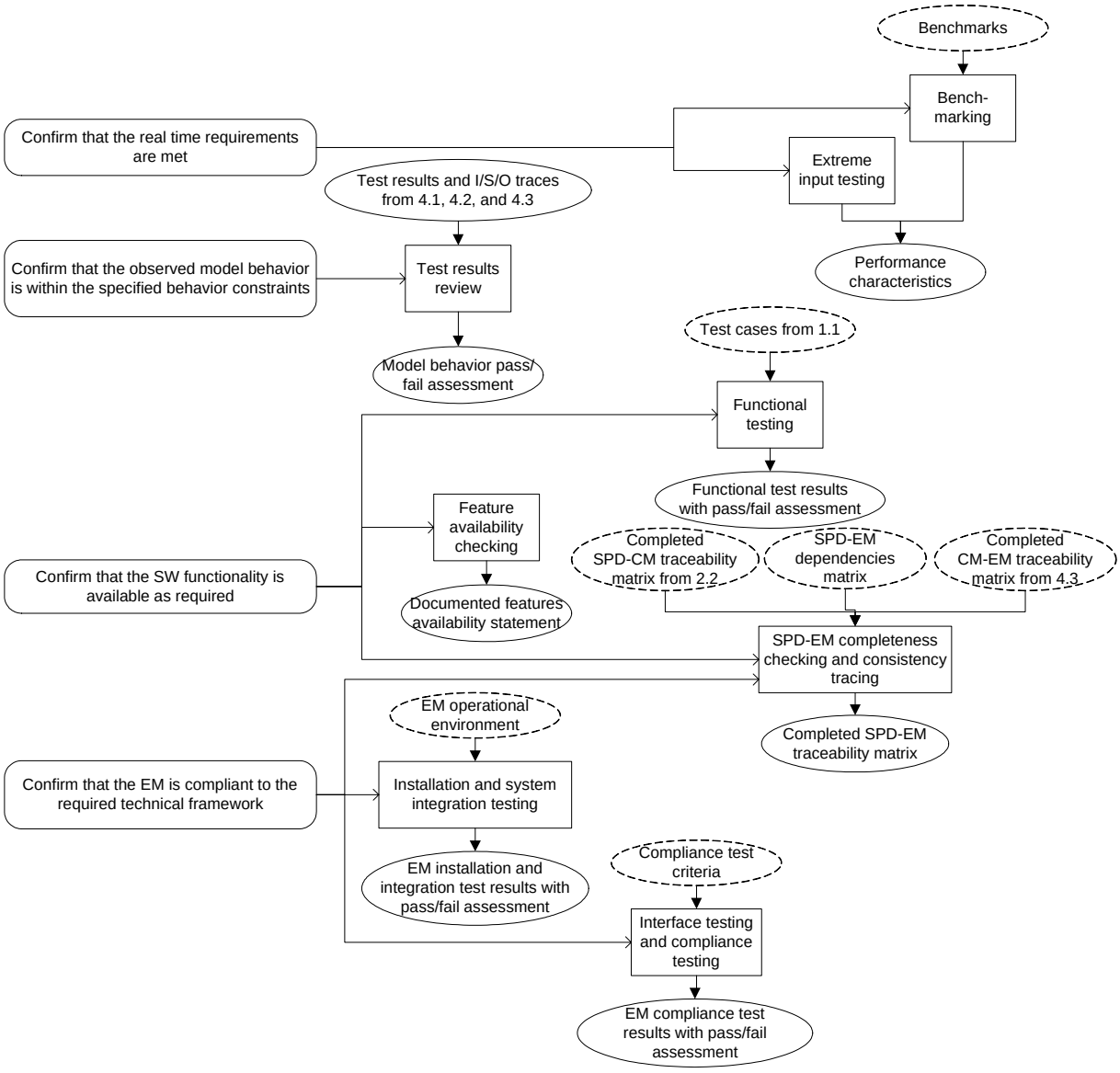


Figure 17: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 4.4

- **Required stage of model development**: Implementation phase completed
- **IP counterpart** (transformation V&V only): SPD

- **Sub-phase goal**: Confirm completeness of the EM in content and consistency with the SPD. The focus of this sub-phase lies on the question, whether the EM satisfies all explicitly specified requirements. Addressed errors include:
    - Violation of real time requirements
    - Violation of explicit implementation requirements
    - Violation of required I/O parameter accuracy
    - Violation of interactivity and interoperability requirements
    - Violation of software functionality requirements
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. It can be skipped, if it is sure that the CM satisfies all requirements given in the SPD (sub-phase 2.2) and the EM implements the CM correctly and suitably (V&V sub-phase 4.3).
- **Impact of error detection on model development**: Regress by three phases. If it is discovered that the EM does not meet all requirements, model development must (at least partially) be repeated from system analysis.
- **Sub-phase objectives**: Confirm that
    - real time requirements are met;
    - the observed model behavior is within the specified behavior constraints;
    - the software functionality is implemented as required;
    - the EM is compliant with the required technical framework (platform, interactivity, interoperability).
- **Proposed techniques for static analysis**:
    - SPD-EM completeness and consistency checking: By reusing the SPD-CM and CM-EM traceability matrices from V&V sub-phases 2.2 and 4.3, and using a template-based SPD-EM dependency matrix, a trace from each requirement to the associated EM contents is created, and documented in the SPD-EM consistency matrix. Requirements without associated EM contents (and vice versa) indicate a problem.
    - Feature availability checking: The simple existence of a required feature is checked.
- **Proposed techniques for dynamic testing**:
    - Interface testing and compliance testing: It is tested, whether the EM is compliant with the required man-machine- and machine-machine-interfaces specifications.
    - Installation and system integration testing: The EM is executed in its operational environment according to the test cases specified in V&V sub-phase 1.1.
    - Benchmarking: The response behavior of the EM is tested according to a well-defined input pattern and measured.
    - Extreme input (load) testing: To test the responsiveness of the EM under extreme conditions, a high internal load is created and the EM behavior recorded.
    - Functional testing: It is tested, whether the EM supports all required experimentation (stimulation and observation) within the specified experimental framework.
- **Proposed techniques for results evaluation**:
    - Pass/fail evaluation: Quantitative generated data is compared to quantitative pass/fail criteria.
    - Test results and test experiment results review: The results provided during V&V sub-phases 4.1, 4.2., and 4.3 are assessed, if they meet the associated pass/fail criteria.
    - Face validation: see above.

- **External references, leveraged information, and reused V&V output**:
  - Benchmarks
  - Test results from 4.1, 4.2, and 4.3
  - Test cases from 1.1
  - Completed SPD-CM traceability matrix from 2.2
  - SPD-EM dependency matrix
  - Completed CM-EM traceability matrix from 4.3
  - EM operational environment
  - Compliance test criteria
- **Provided outputs**:
  - Performance characteristics
  - Pass/fail assessment
  - Functional test results
  - Documented features availability statement
  - Completed SPD-EM traceability matrix
  - EM installation and integration test results
  - EM compliance test results

5.8 SN to EM Transformation V&V

This section documents the goals of V&V sub-phase 4.5, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 18 visualizes their dependencies.

- **Required stage of model development**: Implementation phase completed
- **IP counterpart** (transformation V&V only): SN
- **Sub-phase goal**: Confirm completeness of the EM in content and consistency with the SN. In this sub-phase the remaining question, whether the EM covers the implicit (never specified) needs and is useful for the solution of the user's problem is examined.
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. If it was shown earlier that the SN are complete and consistent with the SPD (V&V sub-phase 1.2), and that the EM is implemented to its specification given by the SPD (V&V sub-phase 4.4), the V&V activities in this section are redundant. However, due to the difficulty to perfectly express the SN in the SPD and to detect errors in the SPD, other than by examining the completed software, it is very likely that still errors remain.
- **Impact of error detection on model development**: Regress by four phases, model development needs to be (at least partially) repeated from the beginning.
- **Sub-phase objectives**: Confirm that
  - the EM duplicates the behavior of the real system as needed;
  - the EM works with other systems as needed;
  - the EM allows experimentation as needed.
- **Proposed techniques for static analysis**:
  - SN-EM completeness and consistency checking: By reusing the SN-SPD and SPD-EM traceability matrices from V&V sub-phases 1.2 and 4.4, and using a template-based SN-EM dependency matrix, a trace from each need to the associated EM contents is created, and documented in the SN-EM consistency matrix. Needs without associated EM contents (and vice versa) indicate a problem.
- **Proposed techniques for dynamic testing**:

- Application: The “test cases” are real cases of model application, or previously unspecified uses of the model closely related to its real application with known outcome. Failure in accepting all relevant inputs is recorded as well as EM behavior for subsequent evaluation.

- **Proposed techniques for results evaluation**:
  - Visualization and face validation

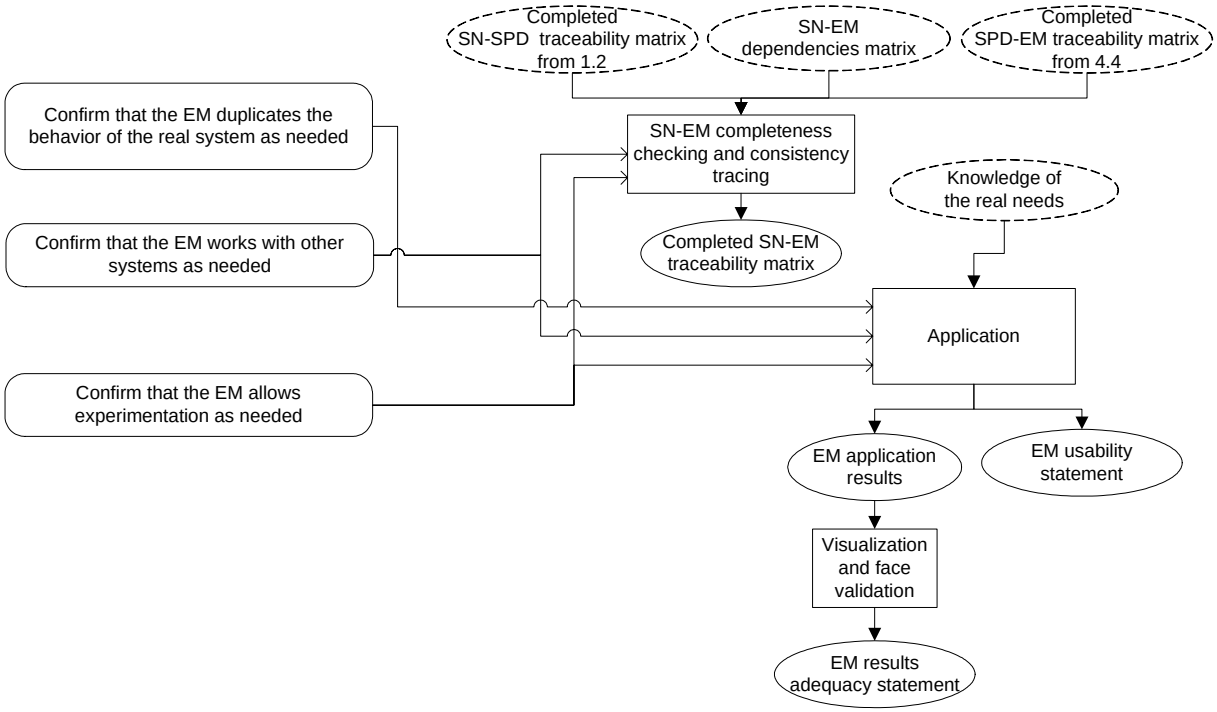


Figure 18: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 4.5

- **External references, leveraged information, and reused V&V output**:
  - Completed SN-SPD traceability matrix from 1.2
  - SN-EM dependency matrix
  - Completed SPD-EM traceability matrix from 4.4
  - Knowledge of the real needs
- **Provided Output**:
  - Completed SN-EM traceability matrix
  - EM application results
  - EM usability statement
  - EM results adequacy statement

6 V&V OF SIMULATION RESULTS

The SR document the observed behavior of the EM, which was stimulated according to the design of experiments and experiment setup over time. They also may include aggregated values created from the observed and recorded model behavior.

6.1 Expected Experimentation Activities

During experimentation the EM is stimulated with input data and produces with advancing simulation time output data depending on its internal state, its input, and (if stochastic simulation is conducted) random elements. The aim of experimentation is to create SR that trigger conclusions for an intended purpose in reality. This implies that the experiments with the EM should be designed in a form that actually allows to do so, which implies that the selection of input data and the design of experiments is an extremely important topic.
When experimenting with an EM, in principal the same mistakes can be made as if experimenting with the real system (although their direct impact is considered to be negligible due to the "unreal" nature of the simulation experiment). Deliberate setup of the experiment(s) is required to allow the desired gain of knowledge or experience. These mistakes are strongly dependent on the application domain from which the overall M&S needs originate and cannot be addressed by a generalized M&S V&V process. However, in addition to these errors in experimentation, there are also numerous mistakes typical for the planning and implementation of simulation experiments, which are addressed in the following.

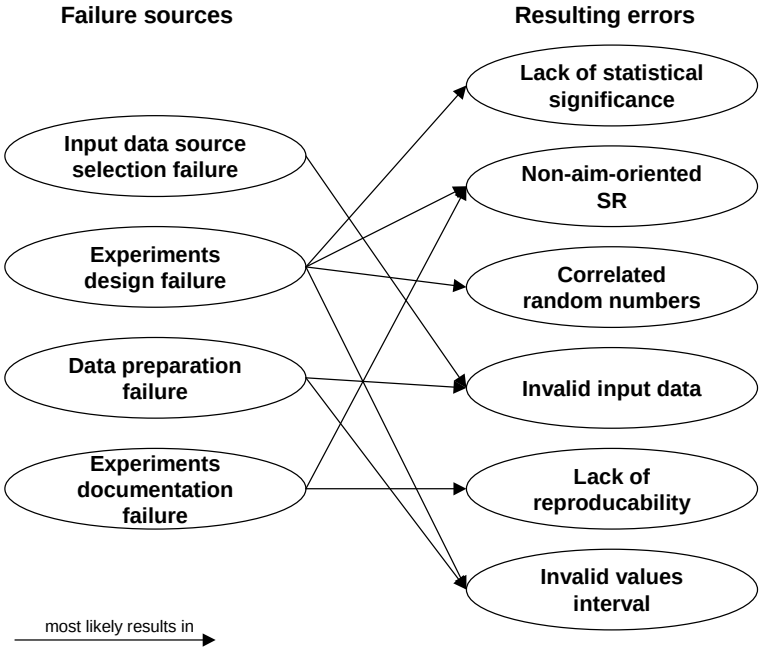


Figure 19: Dependencies between failures in the experimentation phase and errors in the SR

The SR are intended to trigger conclusions concerning the behavior of the real system. They are produced by the user and accepted by the sponsor or a sponsor representative.

- **Input data source selection failure**: During system analysis data is collected that will be used as input data or to generate input data for stimulation of the model. If this data is measured at an unsuitable related system, created by another invalid model, or otherwise calculated in the wrong way, the model output will also be incorrect or unsuitable.

- **Data preparation failure**: Usually, if real input data or embedded data is available, it was recorded during experiments with the real system that were not explicitly done for the purpose of creating information for M&S. This data most probably must be edited before it can be used for stimulation of a model or for comparison to model output. When assessing correctness and suitability of runtime data, it should be transparent how this data was measured and recorded.
- **Experiment design failure**: Besides the creation of an experiment setup (scenario) that does not represent a relevant constellation of real influences on the real system, the randomness incident to stochastic simulation must not be neglected. Any stochastic simulation experiment needs to be repeated sufficiently often to guarantee statistical significance of the observed model behavior. The initialization of the RNG plays an important role; the single simulation runs need to be started with seed values that do not lead to undesired correlation between the random variables values over the different runs.
- **Experiment documentation failure**: Although the experiments are conducted in accordance with all experimentation requirements, they are not documented properly. Meaningful knowledge about the experiments and the SR are lost.

The experimentation phase is re-entered, when the EM or the runtime input data change or an error in the experiment design is detected.

6.2 SR Documentation Requirements

SR may be documented as graphs, tables, data streams (I/S/O traces), or videos; in some cases the visualization of model behavior over time may be relevant for stimulation of human beings (training). SR always include model input data and model output, and should be stored in electronic data files, to allow further computer aided analysis and evaluation. The SR should contain (sample SR are given in Appendix C):

1) **Configuration control information**: Required for configuration control and reference of earlier IP; includes an unique identifier.
2) **Execution environment**: Identification or description of data bases and data collection tools.
3) **Experiment setup**: Description of a set of input combinations under which the EM is executed and the rational for their selection, including
   a) **Aim of experiment**: Description of the desired insight from experimentation.
   b) **Individual simulation run setup**: For each individual simulation run, its input combinations and interaction with external elements need to be recorded. This includes:
      i) the initial state of the EM;
      ii) the model input over time;
      iii) any additional external stimulation, as human interaction or interoperation with black box input data producers.
   c) **Run repetition**: For stochastic simulation, often several repetitions of a run are required. The description includes:
      i) the number of repetitions of each experiment setup;
      ii) the initialization of RNG (seed).
4) **Input data documentation**: (Reference to) description of used input data, including
   a) **Origin/source**: Identification of the data source, e.g., the (related) real system, or another quantitative model;
   b) **Age**: Date of data recording;
   c) **Data collection method and approximated or known measurement error**: Description of the circumstances under which the data was recorded, used tools for data recording, and approximated measurement error;

d) **Data aggregation method**: If the data was aggregated from other quantitative data, besides the above information, also the aggregation method needs to be known.

5) **Documentation of experiment execution**: The documentation of the experiment includes for each executed run
   a) **Documentation of EM behavior (I/S/O trace) over time**: A data record of quantitative model behavior;
   b) **Running speed and response behavior report**: This is of importance, if real time requirements need to be assessed;
   c) **Identification of warm-up period**: In the beginning of the experiment, the internal state of the model may be unrealistic, because, for example, the appropriate number of objects in the system first needs to be instantiated;
   d) **Identification of unusual/undesired/unexpected results**: If there are any exceptions recognized during experimentation, these should be recorded.

There should be a justification for each number no information is given for. The required information may change due to changing simulation aims.

6.3 Potential Errors in the SR

Even with a correct and suitable model, SR may be inadequate to support the solution of a given problem. This may be backtracked to invalid, incorrect, or unsuitable data or inadequate experimentation. There can be errors in the SR; the dependencies between failures during experimentation and errors in the SR are depicted in Figure 19.

- **Invalid runtime data** (internal error): This may be due to inaccurate measurement, measurement at a unsuitable related system, measurements under wrong environmental conditions, wrong derivation of theoretically created input data, fitting the wrong distribution function, or simply out-dated data.
- **Invalid value domain** (internal error): If the input data used is beyond the input value domain the model was build for, it is pure coincidence, if the SR are valid.
- **Lack of statistical significance** (internal error): In a stochastic simulation, the result of a single, too short simulation run often is nearly meaningless. A sufficiently high number of simulation runs must be executed, or if the experiment allows it, its duration needs to be sufficiently long.
- **Lack of reproducibility** (internal error): The SR cannot be reproduced. This implies the assumption that the experiment design is incomplete or inconsistent.
- **Correlated random numbers** (internal error): If the use of the RNG leads to undesired correlations, dependencies in the SR may be detected that are not existent in the real world.
- **Unsuitable or non aim-oriented SR** (transformation error): SR may, although correct, be too far away from the actual problem to help solving it. The experiment setups must be sufficiently close to the real situation to allow conclusions for reality.

The V&V process shall cover all abstract errors identified above. They should be identified in an efficient way as early as possible.

6.4 SR Internal V&V

This section documents the goals of V&V sub-phase 5.1, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 20 visualizes their dependencies.

- **Required stage of model development**: Experimentation completed
- **IP counterpart** (transformation V&V only): N/A

- **Sub-phase goal**: Demonstrate correctness in form and self-consistency of the SR (experiment design and model I/O behavior). This sub-phase aims to ensure that the experiment design (experiments setup, number of experiment repetitions) allows the collection of the required data, and that the observed model output approximates the required real system behavior with sufficient accuracy. Addressed errors include that
    - meaningless experiments are conducted, yielding useless output data;
    - the simulation runs are not harmonized to build a well-designed experiment;
    - the input data is not representative for the real scenario; and
    - observed model output cannot be explained with respect to the (expected) real system behavior.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. If the experiment design does not yield results that contribute to the solution of the user's problem, the usefulness of the complete M&S project must be doubted.
- **Impact of error detection on model development**: Regress by one phase. The experiments have to be redesigned and repeated.
- **Sub-phase objectives**: Demonstrate that
    - the SR (experiment design and the documented model I/O behavior) are correct in form (i.e., consistent with the SR documentation requirements);
    - the SR are internally consistent;
    - the design of the simulation experiments is problem oriented;
    - the input data represents the real environmental conditions with sufficient accuracy; and
    - the model output reflects the real system behavior over time.
- **Proposed techniques for static analysis**:
    - SR documentation completeness and correctness checking: Using a checklist or a template, the experiment design and the model output are checked for completeness and consistency with the template. This results in a completed SR checklist or template.
    - SR documentation internal consistency checking: By use of a template-based SR internal dependency matrix, the SR documentation is checked for internal consistency. This results in a completed SR internal consistency matrix.
    - Compare the simulation experiment setup to a (fictive) real experiment setup: If the setup of the simulation experiment was derived from a real experiment, analysis and comparison of both of them may reveal inadequate differences. Otherwise SMEs are asked, how they would plan the experiments, if they had complete control over the real system.
- **Proposed techniques for dynamic testing**:
    - Monitoring of internal model behavior (instrumentation): Beside the input/output data that is recorded during model execution, additional information (e.g. state variables) about the internal state of the submodels and their interaction is made visible outside and recorded as I/S/O traces.
- **Proposed techniques for results evaluation**:
    - Comparison between real results and the SR: If there are results from a similar real experiment available, they are compared to the SR and analyzed for differences submodel-wise. This can be performed by, e.g., visualization and face validation, a Turing test, or by statistical tests.

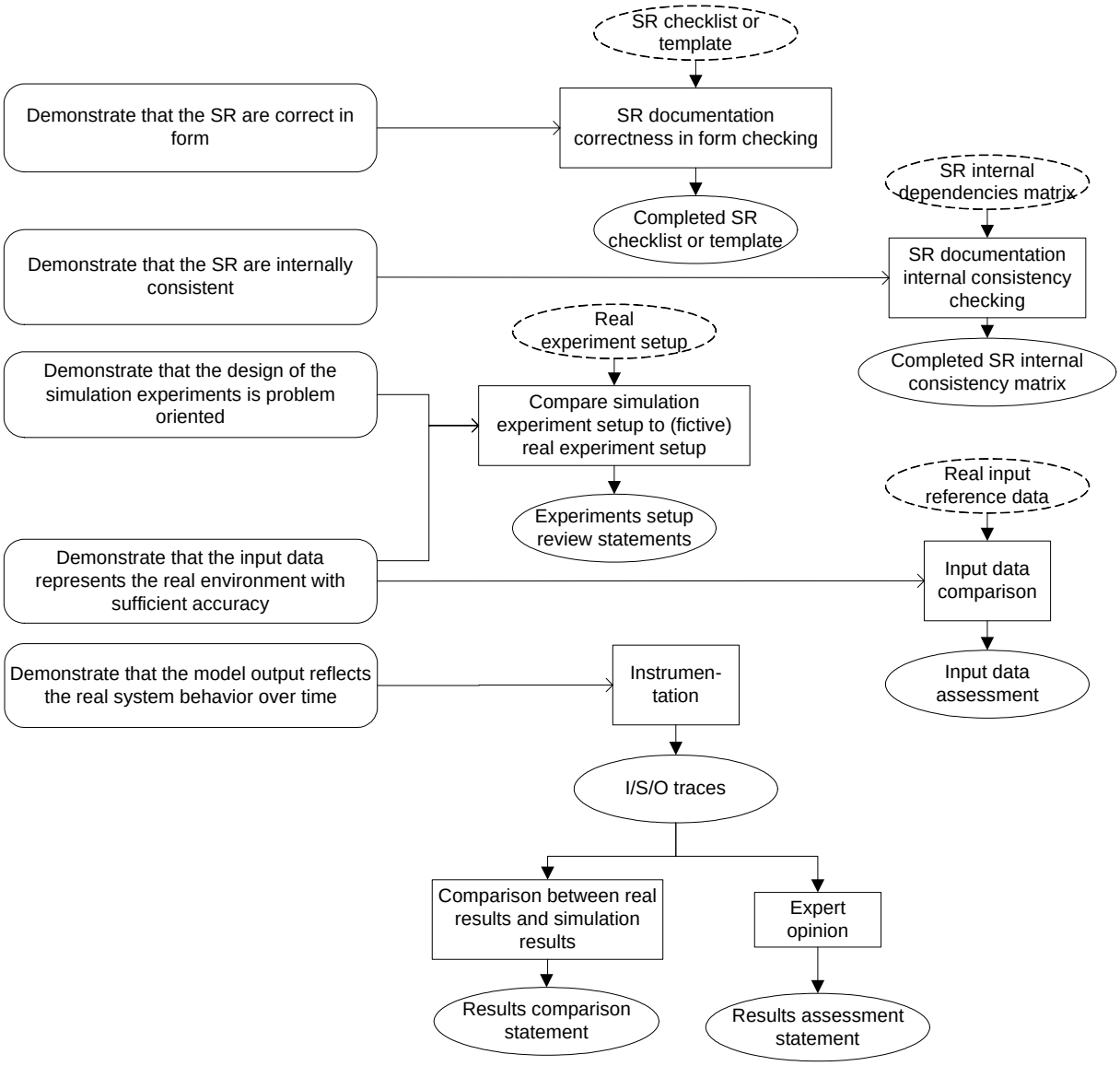


Figure 20: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 5.1

- Expert opinion: Experts judge, whether the simulation data could be the output of a real experiment. This can be achieved via visualization, followed by face validation.
- Input data comparison: The input data used for model stimulation is compared to measured influences or other real input data to the real system. This can be performed by, e.g., visualization and face validation, a Turing test, or by statistical tests.

- **External references, leveraged information, and reused V&V output**:
  - SR checklist or template
  - SR internal dependency matrix
  - Real experiment setup
  - Real input reference data
- **Provided output**:
  - Completed SR checklist or template
  - Completed SR internal consistency matrix
  - Experiment setup review statements
  - Input data assessment

- I/S/O traces
- Results comparison statement
- Results assessment statement

6.5 EM to SR Transformation V&V

This section documents the goals of V&V sub-phase 5.2, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 21 visualizes their dependencies.

- **Required stage of model development**: Experimentation completed
- **IP counterpart** (transformation V&V only): EM
- **Sub-phase goal**: Demonstrate completeness of the SR in content and consistency with the EM. This includes the demonstration that the input data is technically interoperable with the EM, that the experiment design takes the simulation method into account (here: stochastic vs. deterministic simulation), and that the model output is correctly stored in the output data base. Explicitly addressed errors include that
  - an insufficient number of experiment repetitions is executed (stochastic simulation);
  - Byzantine errors occur during data I/O (e.g., little endian vs. big endian);
  - the input data accuracy (number of bytes) is insufficient;
  - EM internal input models are unsuitable representations of the real influences.
- **Requirements for and consequences of sub-phase skipping**: It is not recommended to skip this sub-phase. Experiment design that does not consider the characteristics of the EM (e.g., the simulation method) is at best inefficient, and yields misleading SR at worst.
- **Impact of error detection on model development**: Regress by one phase. The experiments need to be redesigned giving consideration to the simulation method chosen for the EM and executed again.
- **Sub-phase objectives**: Demonstrate that
  - all input data is provided and all output data is recorded in a correct and suitable data format;
  - the experiment design takes the simulation method into account; and
  - the SR are reproducible.

- **Proposed techniques for static analysis**:
  - EM-SR completeness and consistency checking: By using a template-based EM-SR dependency matrix, a trace from each input, state, and output parameter to the associated recorded data items is created, and documented in the EM-SR consistency matrix. This also includes a “simulation method – experiment design consistency check”, as stochastic models require different experiment designs than deterministic models. The simulation runs for stochastic models must be sufficiently often repeated (with different seeds for the random number generator) to produce SR with statistical significance, or stay in steady state for a sufficiently long time. Also needs to be ensured that a self-driven model is not fed with a data trace, and a trace driven model gets more input than just the initialization data.
- **Proposed techniques for dynamic testing**:
  - Internal data representation monitoring: Input data read from an input stream or a configuration file is made visible by the EM for a reviewer, who assesses the suitability of the data reading / writing procedure.

- Input data interval assertions: It is automatically checked, whether the input data lies within the valid input range. Extreme distortions during data reading may be detected this way (e.g., sign errors).
- Experiment duplication: The experiments setup documentation is analyzed and the experiment repeated. If it is not possible to reproduce the previously recorded EM I/O behavior, a problem is detected.

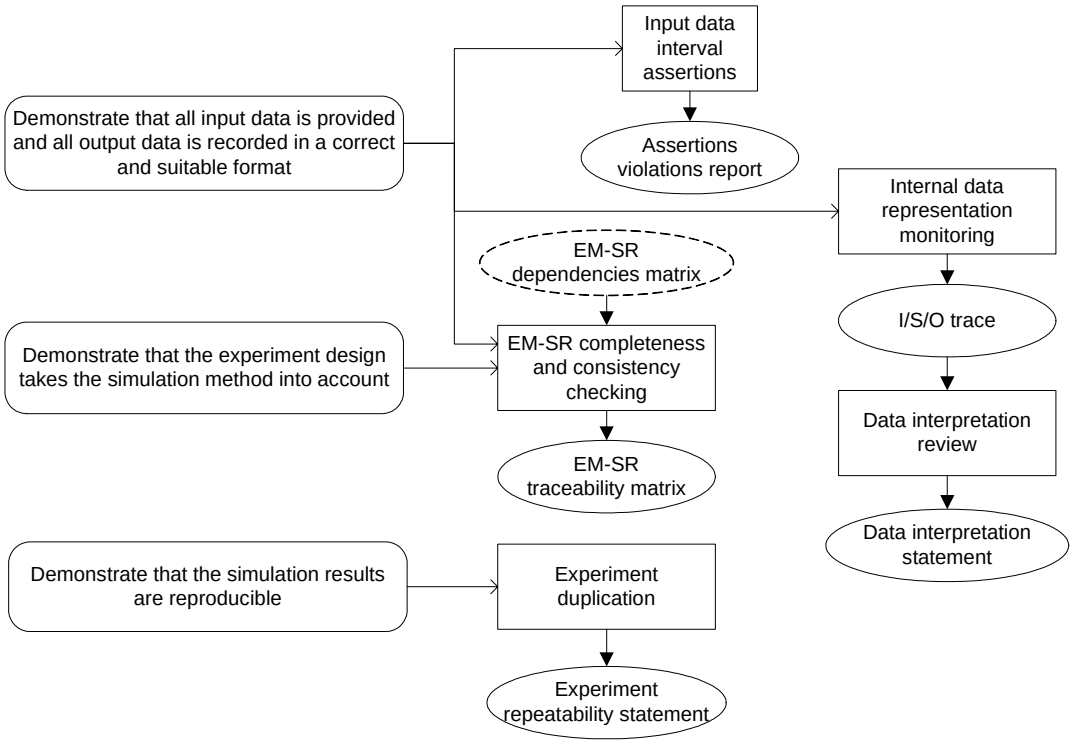


Figure 21: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 5.2

- **Proposed techniques for results evaluation**:
  - Data interpretation review: The monitored input data is compared to the input data in the data base. Differences indicate a problem.
- **External references, leveraged information, and reused V&V output**:
  - EM-SR dependency matrix
- **Provided output**:
  - Assertion violation report
  - I/S/O trace
  - EM-SR traceability matrix
  - Data interpretation statement
  - Experiment repeatability statement

6.6 FM to SR Transformation V&V

This section documents the goals of V&V sub-phase 5.3, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 22 visualizes their dependencies.

- **Required stage of model development**: Experimentation completed
- **IP counterpart** (transformation V&V only): FM

- **Sub-phase goal**: Confirm completeness of SR in content and consistency with the FM. This includes the demonstration that distribution functions for self-driven models are suitable representations of the real external influences. Explicitly addressed errors include that:
    - the input data used for initialization and configuration of the internal data generators (distribution functions) results in an unsuitable representation of the real external influences.
    - the EM internal distribution functions do not reflect the (empirical) distribution functions of the scenario data.
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. If it was previously ensured that the data is preprocessed in the FM as assumed for the experiment design, this sub-phase can be skipped. (This is most likely the case, if the data was recorded during the same system analysis activities, which also are the foundation of the CM. However, if a new experiment was created, or data from another source is used, serious problems can be revealed during this sub-phase.)
- **Impact of error detection on model development**: Regress by two phases. The EM has to be checked for specification violation (partial repetition of software development). Implementation and experimentation need to be repeated.

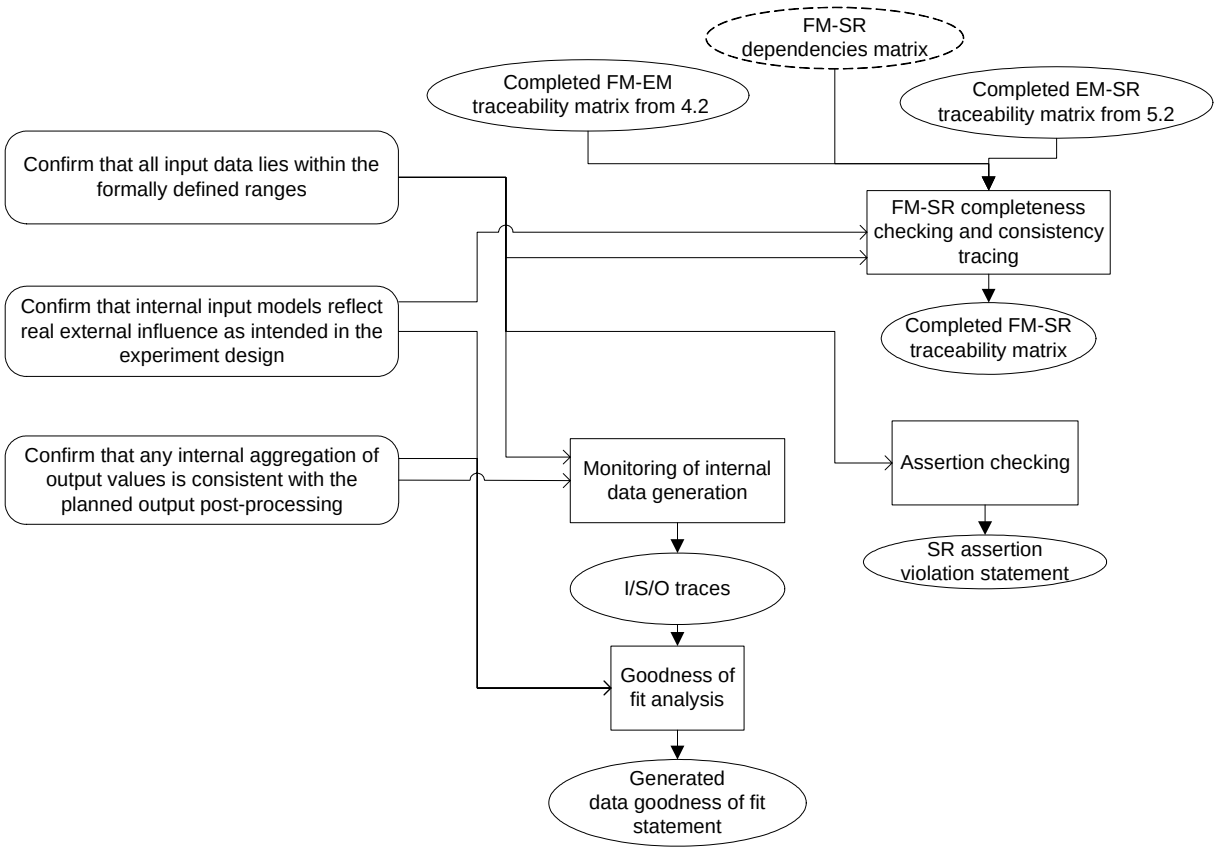


Figure 22: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 5.3

- **Sub-phase objectives**: Confirm that
    - all input data lies within the formally specified ranges;
    - internal input models (e.g., distribution functions) reflect real external influences as intended in the experiment design (consistency between distribution functions);

    - any internal aggregation of output values is consistent with the planned output post-processing.
- **Proposed techniques for static analysis**:
    - FM-SR completeness and consistency checking: By reusing the FM-EM and EM-SR traceability matrices from 4.2 and 5.2, and using a template-based FM-SR dependency matrix, a trace from each formally specified model input and output to the associated data items is created, and documented in the FM-SR consistency matrix. Formally specified model inputs and outputs without associated data items (and vice versa) indicate a problem.
- **Proposed techniques for dynamic testing**:
    - Monitoring of internal data generation (instrumentation): During experimentation with a self-driven stochastic model, after reading the distribution parameters from the data base all internally generated data is made available, and all output data is monitored prior to aggregating it by statistical means.
    - Assertion checking: The input parameters are secured with assertions that only accept input data within the acceptable intervals as defined in the FM.
- **Proposed techniques for results evaluation**:
    - Goodness of fit analysis: By implementation of statistical tests it is examined, whether the internally generated data fit the FM internal distribution functions.
- **External references, leveraged information, and reused V&V output**:
    - Completed FM-EM traceability matrix from 4.2
    - FM-SR dependency matrix
    - Completed EM-SR traceability matrix from 5.2
- **Provided output**:
    - Assertion violation statement
    - Goodness of fit statement
    - I/S/O traces
    - Completed FM-SR traceability matrix

6.7 CM to SR Transformation V&V

This section documents the goals of V&V sub-phase 5.4, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 23 visualizes their dependencies.

- **Required stage of model development**: Experimentation completed
- **IP counterpart** (transformation V&V only): CM
- **Sub-phase goal**: Confirm completeness of the SR in content and consistency with the CM. It should be ensured that input data semantics and input parameter semantics are identical. Inconsistent units (e.g., miles vs. kilometers) or inconsistencies between coordinate systems are typical errors that are detected during this sub-phase.
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. If it is sure that the CM border is correctly and suitably formalized (V&V sub-phase 4.2), and that the input data shows the formal characteristics as required for the FM (V&V sub-phase 5.3), this sub-phase can be skipped. The input values are most likely to be interpreted by the EM as intended during their recording or generation, if they originate form the same system analysis effort as the CM. However, if the input data comes from a different source, the assessment of consistency between model input parameter semantics and input data semantics is essential for the credibility of the SR.

- **Impact of error detection on model development**: Regress by three phases. Partial repetition of the model development phases from model formalization through experimentation.
- **Sub-phase objectives**: Confirm that
  - the EM internal semantic interpretation of the input data values is consistent with the semantics of the recorded or generated input data;
  - the input data intervals do not exceed the input data interval for, which the underlying model was created;
  - the semantics of the output given in the experimental framework (which later will be used for interpretation) is consistent with the EM output parameter semantics;
  - the EM behaves as intended with the CM during all experiments.
- **Proposed techniques for static analysis**:
  - CM-SR completeness checking and consistency tracing: By reusing the CM-FM and FM-SR traceability matrices from V&V sub-phases 3.2 and 5.3, and the use of a template based CM-SR dependency matrix, a trace from each CM input and output parameter to the associated contents of the experiment description or recorded model output is created, and documented in the CM-SR traceability matrix. CM contents without associated SR contents (and vice versa) indicate a problem.
  - Data recording report review: It is analyzed, whether the instrumentation of the real system and the accuracy of the measurement devices allow the recording of data with the same semantics as the input parameters of the model.
- **Proposed techniques for dynamic testing**:
  - Monitoring of internal data interpretation (instrumentation): During experimentation, all input data is made available as interpreted by the EM after reading it from the data base, and all output data is monitored prior to writing it to the data base. The behavior of the executable submodels is recorded as I/S/O traces.
  - Assertion checking: Using the behavior propositions identified during V&V sub-phases 2.1 and 2.2, the input parameters are secured with assertions that only accept input data within the defined intervals.
- **Proposed techniques for results evaluation**:
  - Visualization and face validation: The modeler judges, whether the model behaves with the input data as intended.
  - Automated behavior propositions violation detection: The I/S/O traces are post-processed by language acceptance automata, which only accept those I/S/O traces that are compliant with their underlying behavior propositions.
- **External references, leveraged information, and reused V&V output**:
  - Completed CM-FM traceability matrix from 3.2
  - CM-SR dependency matrix
  - Completed FM-SR traceability matrix from 5.3
  - Data recording report
  - CM behavior propositions from 2.1 and 2.2
- **Provided output**:
  - Assertion violation report
  - Completed CM-SR traceability matrix
  - Data suitability statement
  - I/S/O trace
  - I/O behavior suitability statement
  - Proposition violation statement

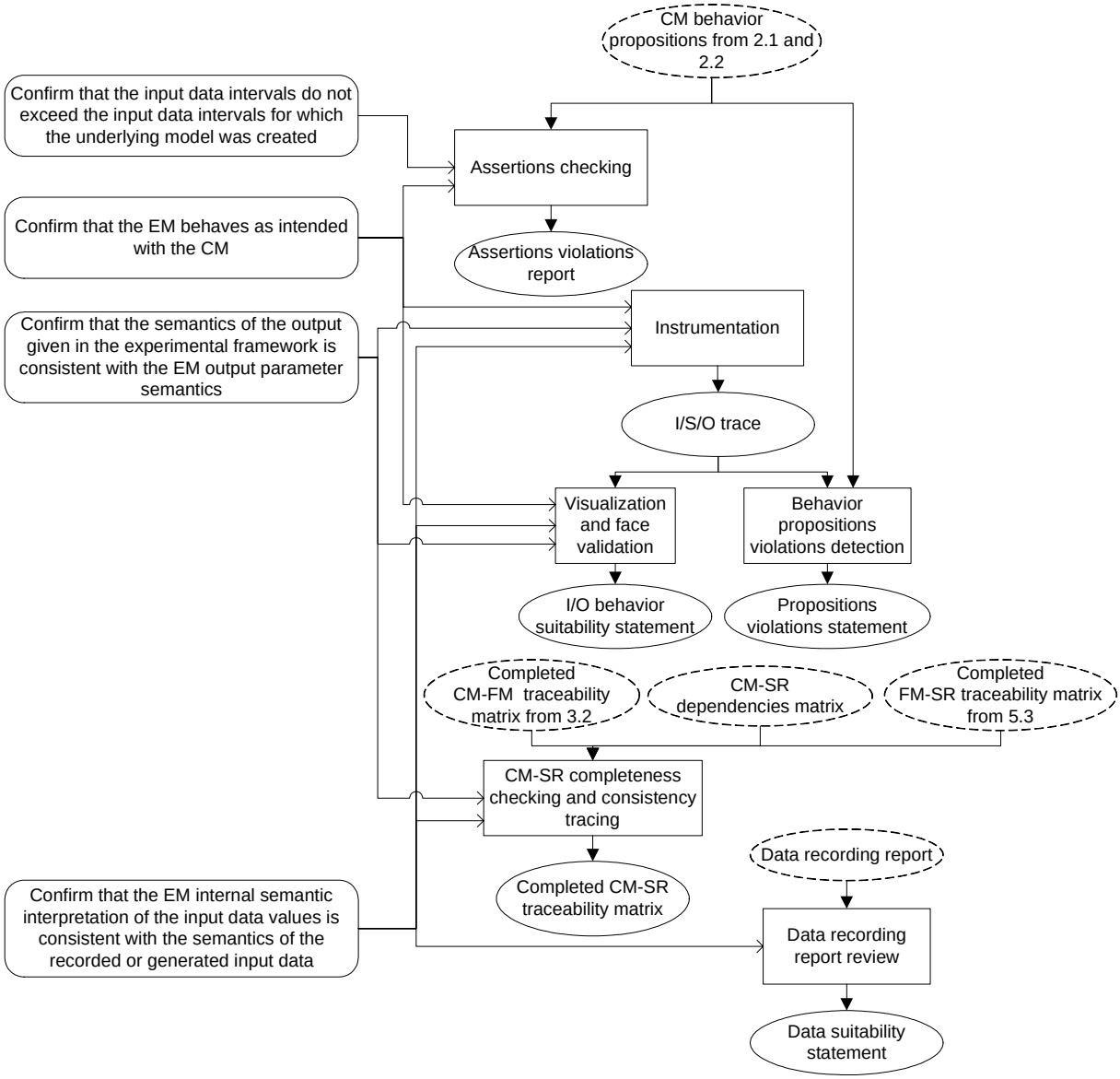


Figure 23: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 5.4

6.8 SPD to SR Transformation V&V

This section documents the goals of V&V sub-phase 5.5, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 24 visualizes their dependencies.

- **Required stage of model development**: Experimentation completed
- **IP counterpart** (transformation V&V only): SPD
- **Sub-phase goal**: Confirm completeness of SR in content and consistency with the SPD. The focus of V&V activities lies on the confirmation that all required experiments were executed according to their specifications, using suitable data for a suitable model. Explicitly addressed errors include experiment designs that deviate from the required experimental framework.
- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. If it was previously shown that the CM (including its experimental frame-

work) satisfies the requirements stated in the SPD (V&V sub-phase 2.2), and that the semantics of the CM border and the actually implemented experimental framework are consistent (V&V sub-phase 5.4), this sub-phase may be skipped.

- **Impact of error detection on model development**: Regress by four phases. If the model results do not satisfy the requirements of the SPD, an error occurred between system analysis and experiment.
- **Sub-phase objectives**: Confirm that all required experiments were executed according to their specifications.
- **Proposed techniques for static analysis**:
  - SPD-SR completeness checking and consistency tracing: By reusing the SPD-CM and CM-SR traceability matrices from V&V sub-phases 2.2 and 5.4, and by using a template based SPD-SR dependency matrix, a trace from each required experiment to the associated contents of the experiment description or recorded model output is created, and documented in the SPD-SR traceability matrix. Experiments required in the SPD without associated SR contents (and vice versa) indicate a problem.
  - Experiment review: Differences between the required experiments and the conducted experiments are analyzed and evaluated.
- **Proposed techniques for dynamic testing**: N/A
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
  - Completed SPD-CM traceability matrix from 2.2
  - SPD-SR dependency matrix
  - Completed CM-SR traceability matrix from 5.4
- **Provided output**:
  - Experiment compliance statement
  - Completed SPD-SR traceability matrix

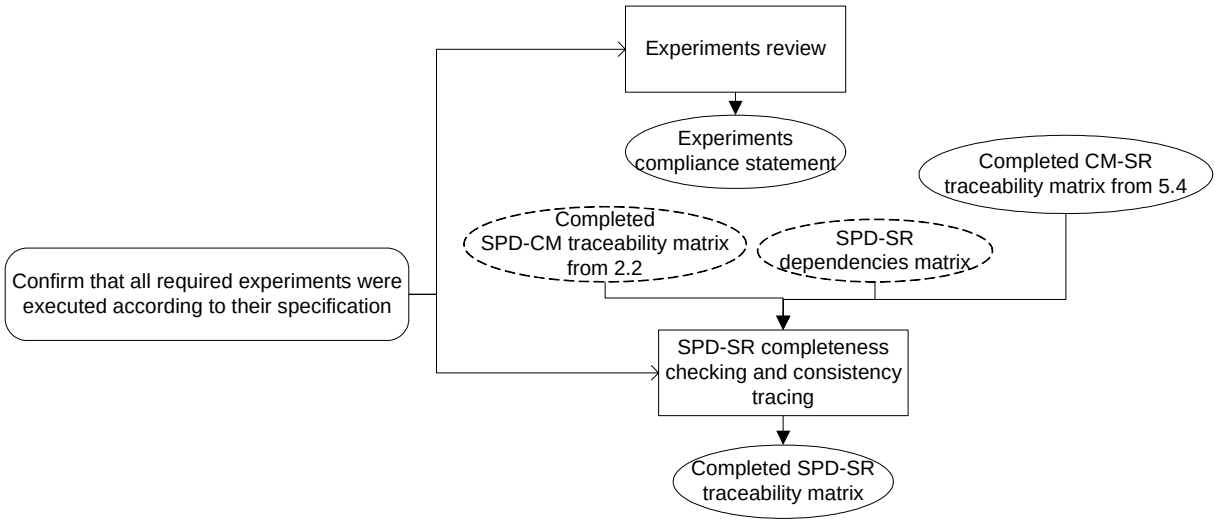


Figure 24: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 5.5

6.9 SN to SR Transformation V&V

This section documents the goals of V&V sub-phase 5.6, potential errors explicitly searched for, derived V&V objectives, proposed V&V techniques, their required inputs, and the outputs (V&V evidence) provided by them. Figure 25 visualizes their dependencies.

- **Required stage of model development**: Experimentation completed
- **IP counterpart** (transformation V&V only): SN
- **Sub-phase goal**: Confirm completeness of the SR in content and consistency with the SN. This sub-phase focuses mainly on the confirmation that the SR contribute to the solution of the user's problem.

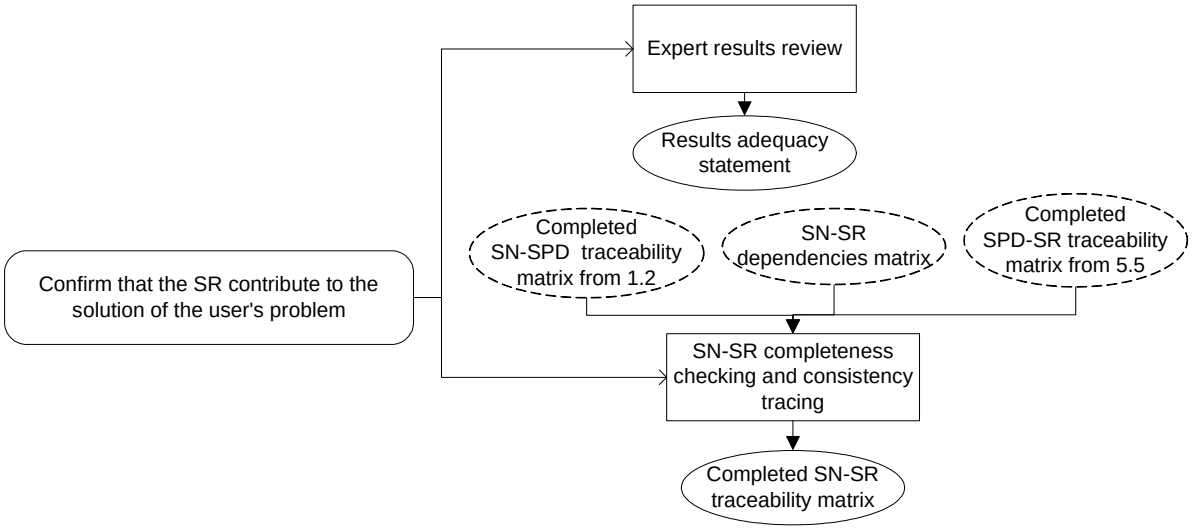


Figure 25: Dependencies between the objectives, techniques, inputs, and outputs of V&V sub-phase 5.6

- **Requirements for and consequences of sub-phase skipping**: This is a confirmation sub-phase. As long as the SPD documents the SN (which was demonstrated in V&V sub-phase 1.2), and the SR satisfy the SPD (V&V sub-phase 5.5), this sub-phase can be skipped. Its objectives should definitely have been addressed during earlier V&V activities.
- **Impact of error detection on model development**: (Partial) repetition of the whole model development process.
- **Sub-phase objectives**: Confirm that the SR contribute to the solution of the user's problem.
- **Proposed techniques for static analysis**:
  - SN-SR completeness checking and consistency tracing: By reusing the Sn-SPD and SPD-SR traceability matrices from V&V sub-phases 1.2 and 5.5, and by using a template based SN-SR dependency matrix, a trace from each desired insight into the real system to the associated contents of the experiment description or recorded model output is created, and documented in the SN-SR traceability matrix. Needed insights without associated SR contents (and vice versa) indicate a problem.
  - Expert review: The experiment design and the model output are reviewed for plausibility. If M&S results contradict other available "conventional" knowledge, an explanation for this contradiction should be found.
- **Proposed techniques for dynamic testing**: N/A
- **Proposed techniques for results evaluation**: N/A
- **External references, leveraged information, and reused V&V output**:
  - Completed SN-SPD traceability matrix from 1.2
  - SN-SR dependency matrix
  - Completed SPD-SR traceability matrix from 5.5

- **Provided output**:
  - Results adequacy statement
  - Completed SN-SR traceability matrix

7 INDEX OF FIGURES

Figure 1: Dependencies between failures in the Problem Definition Phase and errors in the SPD 3
Figure 2: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 1.1 7
Figure 3: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 1.2 9
Figure 4: Dependencies between failures in the System Analysis Phase and errors in the CM 11
Figure 5: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 2.1 16
Figure 6: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 2.2 19
Figure 7: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 2.3 21
Figure 8: Dependencies between failures in the Formalization Phase and errors in the FM 23
Figure 9: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 3.1 27
Figure 10: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 3.2 29
Figure 11: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 3.3 31
Figure 12: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 3.4 33
Figure 13: Dependencies between failures in the Implementation Phase and errors in the EM 35
Figure 14: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 4.1 39
Figure 15: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 4.2 43
Figure 16: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 4.3 45
Figure 17: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 4.4 46

Figure 18: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 4.5 49
Figure 19: Dependencies between failures in the Experimentation Phase and errors in the SR 50
Figure 20: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 5.1 54
Figure 21: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 5.2 56
Figure 22: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 5.3 57
Figure 23: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 5.4 60
Figure 24: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 5.5 61
Figure 25: Dependencies between the objectives, methods, inputs, and outputs of V&V sub-phase 5.6 62

Appendix B: The V&V Triangle Overview

	SPD	CM	FM	EM	SR
SR					(5.1) Demonstrate that • the SR (experiment design and the documented model I/O behavior) are correct in form (i.e., consistent with the SR documentation requirements); • the SR are internally consistent; • the design of the simulation experiments is problem oriented; • the input data represents the real environmental conditions with sufficient accuracy; and • the model output reflects the real system behavior over time.
EM				(4.1) Demonstrate that • the EM, its supplemental information, and the implemented experimental framework are correct in form (i.e., consistent with the EM documentation requirements), • the contents of the EM and its supplemental information are consistent with each other, • the EM is stable, reliable, and testable software (i.e., consistent with Software Engineering "best practice"), • the simulation infrastructure (including time management and random number generation) is consistent with its specification, and • the distributed executable submodels are technically interoperable, if applicable.	(5.2) Demonstrate that • all input data is provided and all output data is recorded in a correct and suitable data format; • the experiment design takes the simulation method into account; and • the SR are reproducible.
FM			(3.1) Demonstrate that • the FM, its supplemental information, and the formally specified experimental framework are correct in form (i.e., consistent with the required FM template) and consistent with each other; • there are no syntax or semantic errors in the FM i.e., the FM is consistent with the modeling formalism specification; • there is no ambiguity in the FM.	(4.2) Demonstrate that • the chosen implementation technique of the simulation infrastructure is suitable for the formalism the FM is based on, • the EM structure is complete and consistent with FM, • the EM implements the internal behavior and interactions completely and consistently with the FM, • all random number distributions are implemented as defined in the FM, and • data types and algorithms are transformed to digitally computable form acceptably	(5.3) Confirm that • all input data lies within the formally specified ranges; • internal input models (e.g., distribution functions) reflect real external influences as intended in the experiment design (consistency between distribution functions); • any internal aggregation of output values is consistent with the planned output post-processing.

	SPD	CM	FM	EM	SR
CM		(2.1) Demonstrate that • the CM (including its supplemental information and the conceptually described experimental framework) is correct in form (i.e., consistent with the required CM template), • all contents of the CM are consistent with each other, • all contents of the CM (including structure, embedded data, anticipated behavior, and experimental framework) are unambiguous and factually correct (consistent with domain knowledge), • the data sources are suitable, and • all data are sufficiently accurate.	(3.2) Demonstrate that • the FM completely and consistently reflects the symbolic structure and behavior description of the CM; • further (formalism enforced) abstraction and idealization is suitable and correct (algorithms, approximation methods, data); • the formally specified experimental framework is complete and consistent with the conceptually described experimental framework of the CM, including input models and distribution functions.	(4.3) Confirm that • the executable submodel structure is consistent with the description of the submodel structure provided in the CM; • the heuristics and algorithms implemented in the EM for internal submodel behavior and submodel interaction, including their further abstraction for implementation due to digitalization, yield behavior data with sufficient accuracy; • the (actually observed) EM behavior is consistent with domain knowledge.	(5.4) Confirm that • the EM internal semantic interpretation of the input data values is consistent with the semantics of the recorded or generated input data; • the input data intervals do not exceed the input data interval for, which the underlying model was created; • the semantics of the output given in the experimental framework (which later will be used for interpretation) is consistent with the EM output parameter semantics; • the EM behaves as intended with the CM during all experiments.
SPD	(1.1) Demonstrate that • the SPD is correct in form (i.e., consistent with the required SPD template); • all contents of the SPD are consistent; • all contents of the SPD are factually correct (i.e., consistent with domain knowledge); • all contents of the SPD are unambiguous; all contents of the SPD are testable.	(2.2) Sub-phase objectives: Demonstrate that • the model border (I/O parameters, their semantics, their accuracy), • the degree of abstraction and idealization, • the model structure and hierarchical decomposition, • the representation of time, • the anticipated model behavior (including embedded data), and • the experimental framework meet the requirements.	(3.3) Confirm that • the FM is complete in content and consistent with the SPD; • the symbolic behavior specification in the FM is as accurate as required; and • the FM is compliant with the required formal framework.	(4.4) Confirm that • real time requirements are met; • the observed model behavior is within the specified behavior constraints; • the software functionality is implemented as required; • the EM is compliant with the required technical framework (platform, interactivity, interoperability).	(5.5) Confirm that all required experiments were executed according to their specifications.
SN	(1.2) Demonstrate that • the model border (I/O parameters, accuracy, update conditions) is defined as needed; • all developmental constraints (conceptual modeling requirements, formalization requirements, implementation requirements) are defined as needed; • the experimental framework is defined as needed; • the specified test cases allow the assessment of acceptability criteria satisfaction.	(2.3) Confirm that • the model border (including anticipated accuracy of parameters), • the degree of abstraction, • the model structure and hierarchical decomposition, • the representation of time, • the model behavior (including embedded data), and • the experimental framework allow the needed experimentation.	(3.4) Confirm that • the model border allows the needed experimentation (stimulation and observation), and • the FM behavior description (including embedded data) is as accurate as needed.	(4.5) Confirm that • the EM duplicates the behavior of the real system as needed; • the EM works with other systems as needed; • the EM allows experimentation as needed.	(5.6) Confirm that the SR contribute to the solution of the user's problem.

Table A: V&V objectives overview

	SPD	CM	FM	EM	SR
SR					• Completed SR checklist or template • Completed SR internal consistency matrix • Experiment setup review statements • Input data assessment • I/S/O traces • Results comparison statement • Results assessment statement
EM				• Completed EM checklist or template • Completed EM internal consistency matrix • Software QA report and products • EM control flow graph • EM object flow graph • EM flow graph test cases and test results • RNG analysis report, random numbers sequences, and RNG statement • EM integration test cases and EM integration test results • Simulation control analysis report • Simulation control test cases • Simulation control test results	• Assertion violation report • I/S/O trace • EM-SR traceability matrix • Data interpretation statement • Experiment repeatability statement
FM			• FM check protocols, token sequences, parse trees • Completed FM checklist or template • Completed FM internal consistency matrix • FM interfaces consistency statement	• Simulation infrastructure analysis statement • Simulation infrastructure test cases • Simulation infrastructure test results • EM results and I/S/O traces • EM-FM results comparison statement • Completed FM-EM consistency matrix • Flow graph comparison statement • Distribution generator analysis statement • Goodness of fit statement • Interface analysis statement	• Assertion violation statement • Goodness of fit statement • I/S/O traces • Completed FM-SR traceability matrix

	SPD	CM	FM	EM	SR
CM		• Completed CM checklist or template • Completed CM internal consistency matrix • Extracted behavior propositions for each submodel • Additional CM test cases • Interpreted CM behavior record • CM structure factual correctness statements • CM assumptions catalogue • Data suitability statements	• Completed CM-FM traceability matrix • FM object flow graphs and comparison statements • FM control flow graphs and comparison statements • FM assertions violations report • FM behavior propositions violations report • FM interpretation results • FM analytical solution results • FM flow graphs review statement	• Completed EM-CM traceability matrix • Assertion violations report • Documented sensitivity plausibility statement • Flow graph comparison statement • EM I/S/O traces • EM behavior plausibility statement • EM behavior propositions violation statement	1) Assertion violation report • Completed CM-SR traceability matrix • Data suitability statement • I/S/O trace • I/O behavior suitability statement • Proposition violation statement
SPD	• Completed SPD checklist or template, including correctness statement • Completed SPD internal consistency matrix • SPD factual correctness statement • SDP test cases	• Completed SPD-CM traceability matrix • CM structure compliance statement • Additional CM test cases • Additional CM behavior propositions • CM behavior compliance statement • CM assumptions compliance statement	2) Completed SPD-FM traceability matrix • Additional FM interpretation results • Additional FM analytical solution results • FM behavior requirements compliance statement	• Performance characteristics • Pass/fail assessment • Functional test results • Documented features availability statement • Completed SPD-EM traceability matrix • EM installation and integration test results • EM compliance test results	• Experiment compliance statement • Completed SPD-SR traceability matrix
SN	• SPD adequacy statement • Completed SN-SPD traceability matrix • Additional test cases with predicted outcomes and pass/fail criteria	• Completed CM-SN traceability matrix • Additional CM interpreted behavior (oracles) • Behavior propositions adequacy statement • CM structure adequacy statement • CM behavior adequacy statement • CM assumptions adequacy statement	• Completed SN-FM traceability matrix • FM flow graph adequacy statement • Additional FM test cases • Additional FM results • FM behavior adequacy statement	• Completed SN-EM traceability matrix • EM application results • EM usability statement • EM results adequacy statement	• Results adequacy statement • Completed SN-SR traceability matrix

Table B: V&V evidence overview

APPENDIX C

"AN DER POINT SÜD"

This appendix documents the Intermediate Products generated during an M&S sample project. The model is not a result of a real road traffic simulation study, but a simplified example to demonstrate the application of the V&V Triangle. All questions are pure fiction; however, a real intersection with real hard measured data was used. The model development and execution was documented in accordance with the documentation requirements for Intermediate Products provided in the main document.

1 SPONSOR NEEDS

(Documented according to the Sponsor Needs template in Appendix A of the main document.)

1.1 Configuration Control

Version Identifier: ADP-SN-V1

1.2 Purpose and Background

1.2.1 General Problem Set

Individual traffic increases more quickly than the expansion of the traffic infrastructure, which leads especially in large cities to traffic jams or traffic breakdowns during rush hours. As constructional changes of the road network usually are expensive and most often possible only in limited form, the potentials and limits of different traffic control policies shall be explored.
Usually, the intersection between "An der Point Süd" and the Autobahn exit and entrance "München Riem" to BAB 94 is subjected only to light traffic. However, during trade shows at the "Neue Messe München", peak load is created that requires a more efficient handling of vehicle traffic.

1.2.2 Examination Aim

Several policies for controlling the lights at the affected intersection are available (Appendix C.1), but the performance of the intersection operated with different light timing policies is unknown. Observations of the queuing behavior of vehicles in heavy traffic situations are required to support the selection of the most appropriate lights control strategy.

1.2.3 Motivation for Modeling and Simulation

Applying the different policies for lights control for a representative period of time at the intersection, and reprogramming of the lights control device is more expensive than running a simulation study. Usually traffic is too low to allow meaningful evaluation, and during trade shows, experimentation may result in undesired, unnecessary traffic obstruction.

1.2.4 Modeling and Simulation Aim and Purpose

The aim of M&S is to reflect the vehicles behavior in the intersection, to allow simulation of the traffic flow under application of selected policies for lights control. The aim of simulation is to duplicate the dynamic flow of traffic over time. The purpose of M&S is to gain quantitative performance data for the different policies, to support a more objective identification of the most appropriate lights timing policy.

1.3 Scenario Description

1.3.1 Idealized General Description

Vehicles approach the lights choosing a lane appropriate for their destination. The incoming vehicles on each lane are counted by electromagnetic induction loops. The vehicle rate per minute is recorded in a data base. Vehicles ac- and decelerate as required, avoiding collisions with other vehicles according to the traffic rules defined by German law, and wait when facing red lights. The length of each queue is recorded frequently.

1.3.2 Influence Parameters

- The *intensity of traffic* needs to be considered. It is given by the mean arrival rate of vehicles for each incoming lane, which is available as [#vehicles / 60s].
- The *switching behavior* of the traffic lights controls the decrease of queue lengths. It is defined by the switching logic of lights control, which is available as a table containing the durations and order of lights colors.

1.3.3 Goal Parameters

- *Queue length* is observed. It is defined by the number of vehicles on each lane in front of a lights, the length of each vehicle [m], and the space between them [m], yielding the total length of the queue in [m].
- *Waiting time* of a vehicle in the intersection is defined as the duration of time during which the vehicle moves slower than 1m/s, and is measured in [s]. It shall be recorded for each vehicle.

1.3.4 Acceptable Limitations

The examination of rare traffic situations in the intersection is not intended, therefore a fault-free operation mode of the lights may be assumed. Accidents or road constructions that block one or more lanes in the intersection do not need to be considered. It is assumed that all traffic participants behave properly and do not cause crashes. Critical weather conditions that may influence the behavior of the vehicles are out of scope of the examination. In addition, lane changing behavior and overtaking behavior are excluded during the experiment. It is assumed that all vehicles have chosen their appropriate lane prior to entering the intersection area.

1.4 Operation Environment

1.4.1 Operation Platform and Operating System

Personal Computer with Windows NT 4.0, or higher.

1.4.2 Interoperability

No requirements, a stand alone solution is acceptable.

1.4.3 Man-Machine-Interfaces

No interactivity is required; the simulation output shall be available after the simulation execution in text files.

1.5 Worst Case Impacts Statement

When applying the strategy for lights control identified as most appropriate during simulation, the impact on traffic has to be observed. No safety related measures are modified, the danger for human health is assumed to be negligible. Minor economic damage may be caused by increase of jamming. This will be resolved by returning to the previously established lights control strategy. The expenses for the simulation study are negligible. The worst case impact of failure of this simulation study is considered to be low.

2 STRUCTURED PROBLEM DESCRIPTION

(Documented according to the Structured Problem Description template in Appendix A of the main document.)

2.1 Configuration Control

Version Identifier: ADP-SPD-V1, based on ADP-SN-V1

2.2 Sub-Aims Hierarchy

Overall aim: Analyze the traffic flow in the intersection under consideration and variation of the intensity of traffic and the lights control strategy.

- Sub-aim 1: Observe the dynamic behavior of vehicle queues over time, extract the statistical queue length characteristics.
- Sub-aim 2: Observe the dynamic flow of each individual vehicle and its waiting time over time, extract statistical waiting time characteristics for all vehicles.

2.3 Model I/O Requirements

2.3.1 Conceptual Description

2.3.1.1 Consider the following influence parameters

- *Lights switching table*: This table controls the (cyclic) switching of lights over time. This input is relevant, as the influence of the lights control strategy on traffic is subjected to examination. The lights control table may contain all allowed lights combinations, but must comply to the law regulations for traffic lights switching. (E.g., the duration of the yellow phase may not be changed, and the duration for "red in both intersecting directions" must not fall below the allowed minimum). Lights control strategies applied for simulation must be approved as acceptable strategies separately.
- *Vehicle inter arrival times* for each direction and lane: This influence parameter controls the density of traffic in each direction. The density of traffic must be considered when judging the appropriateness of a lights control strategy. The unit of the arrival rate is [#vehicles/time interval]. It may never be negative, and its maximum is limited by the maximum speed observed and the space each vehicle occupies.

2.3.1.2 Allow observation of the following goal parameters

- *Queue length*: This parameter reflects samples of the number of vehicles on the lanes before passing the traffic lights. The sampling rate may be varied. This parameter is required to observe the simulated dynamic change of queue lengths. An estimate of the queue length in [m] can be created from the number of vehicles, their estimated lengths and the accumulated length of the spaces in between them.
- *Waiting times* of vehicles: Reflects the accumulated time a vehicle spends waiting at red lights or in the intersection in [s]. This parameter is required to judge the impact of the lights switching strategy on the drivers. As time is experienced by drivers subjectively and relatively, accuracy requirements are average.

2.3.2 Formal Specification

The mean inter arrival time shall be used as a parameter for an appropriate statistical distribution function to generate input streams of vehicles.

2.3.3 Technical Implementation

The modification of the input parameters *lights switching table* and *vehicle inter arrival times* shall be simple; configuration files are preferred. The desired output values shall be recorded in txt-files that can be read with any text editor.
Data exchange during run-time with other models, hardware systems, or a man-machine interface is not required.

2.4 Model Structure and Behavior Requirements

2.4.1 Conceptual Description

2.4.1.1 A priori identified structural elements

- *Vehicles*: These are the atomic units for the desired observations. The traffic flow depends on the behavior of the individual vehicles. Their generation depends on the vehicle arrival rate.
- *Traffic Lights*: Main control instance for the traffic flow. The influence of the modification of the lights behavior is subjected to examination. Their switching behavior will be controlled by the lights switching table.
- *Intersection geometry*: The impact of the queues on the remaining traffic (e.g., blocking lanes) depends on the queue lengths and the geometry of the intersection. The intersection geometry will not be manipulated during a simulation run.
- *Pedestrians (ignore!)*: The number of pedestrians in the observed intersection area usually is negligible and does not impact the flow of vehicles.

The behavior of the structural elements identified above shall be analyzed and appropriately modeled.

2.4.1.2 Imported structural elements

No structural elements are imported form other models, nor are other data models used.

2.4.1.3 A priori identified cause effect dependencies

The following cause-effect dependencies shall be considered in the model:

- red lights cause vehicles to stop;
- slower predecessors cause vehicle to decelerate to predecessor's velocity;
- narrow curves cause vehicles to decelerate;
- free road causes vehicles to accelerate to maximum desired velocity.

2.4.1.4 A priori identified behavioral constraints

There is no overtaking in the intersection area.

2.4.2 Formal Specification

For the formal specification of the structural dependencies, the appropriate UML diagrams, DEVS, or Z shall be used. The behavior of the *vehicles* shall be unmistakably defined using kinematic equations. All other behavior is to be specified within the appropriate diagrams of the UML, or Z.

2.4.3 Technical Implementation

To enhance the reuse of encoded structural elements of the model in similar models, an object-oriented approach shall be taken during implementation.

2.5 Experimental Requirements

2.5.1 Conceptual Description

Simulation experiments shall be executed for the intersection under “standard load”. Then the load shall be systematically increased to extremely heavy traffic.

2.5.2 Formal Specification

No formal specification of the experimental design is required.

2.5.3 Technical Implementation

A script file shall allow to automatically repeat experiments, to change the model parameters, and to manage recording of simulation results.
Simulation experiments shall be conducted on a Windows NT4.0 PC, or higher.

3 CONCEPTUAL MODEL

(Documented according to the Conceptual Model template in Appendix A of the main document.)

3.1 Configuration control

Version Identifier: ADP-CM-V1, based on ADP-SPD-V1

3.2 Model Overview

The model "An der Point Süd" is an abstracted and idealized representation of an intersection close to the "Neue Messe München". It allows the observation and examination of intersection performance (vehicle queuing behavior), depending on the chosen lights control policy and the chosen traffic load. For this purpose, the behavior of the individual vehicles in the intersection is modeled.

3.3 Structure and Behavior Description

3.3.1 Overall Model Layer

The description of the input and output parameters of the overall conceptual model is given below. Neither input, nor output comply to any particular standard.

3.3.1.1 Intersection Input

- *Inter arrival time*: Arriving vehicles will be generated within the model during simulation. For each *incoming lane i* an input parameter λ_i must be set that controls the density of each stream of incoming vehicles. The unit of the inter arrival time is [s], allowed values are within the interval [1; ∞], considered accuracy are two decimal places. These parameters are initialized once and do not change over simulation time.
- *Lights control table*: A cyclic switching pattern for the lights is read from a switching table. This table contains the *lights states* with a set of *color codes* [red, redyellow, green, yellow] to control each lights in the intersection, and the *duration* [s] of the state. This table must be initialized at the beginning of a simulation experiment and does not change over simulation time.

3.3.1.2 Intersection Output

- *Inter arrival rate*: Relative frequency of numbers of vehicles actually generated per minute for each lane. This is an output parameter for control purposes, as the inter arrival time (input parameter) is derived from the inter arrival rates provided in the data files. It will be available during simulation in an interval of 60s of simulation time.
- *Queue length* for each lane: Samples of queue lengths. It will be available during simulation in an interval of 5s of simulation time.
- *Waiting time* per vehicle for each lane: Accumulated time [s] during which the vehicle moves slower than 1 m/s, considered accuracy are two decimal places. This becomes available whenever a vehicle leaves the intersection.

3.3.1.3 Intersection structure description

The overall model *intersection* consists of three sub-models:

- *LightsCtrl*: Controls the switching behavior of the lights.
- *Vehicle*: Reflects a vehicle in the intersection. Multiple instances of the sub-model will exist.

- *Intersection Geometry*: Models the geometric layout (lanes, stopping lines, positions of lights) of the intersection.

The dependencies between the submodels are sketched in Figure 1.

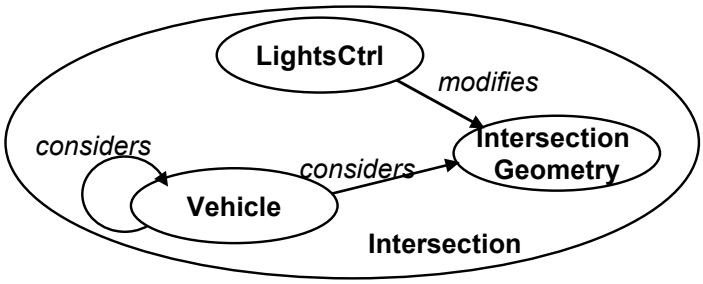


Figure 1: Sketch of the overall conceptual model

3.3.1.4 Intersection behavior description

The behavior of the overall model is defined by the behavior of the submodels on layer 1 and their interaction.

3.3.2 Submodels Layer 1

3.3.2.1 Structure Description of Intersection Geometry

The foundation of the intersection geometry is the CAD drawing provided in Appendix C.2. According to the lanes in the intersection, it is assumed that vehicles follow paths that intersect, fork, or join with other vehicle paths, as depicted in Figure 2. Width information of the lanes is omitted, as only queue lengths are of interest. The pieces of the paths between joining, forking, or intersection points are called “tracks” in the following. The intersection geometry is modeled as a network of tracks, interconnected by nodes. The length of the tracks given in Table 1 was extracted from the CAD drawing, scaled 1:500. Although the CAD drawing is just a faxed copy, it is assumed that the resulting scale error is negligible.

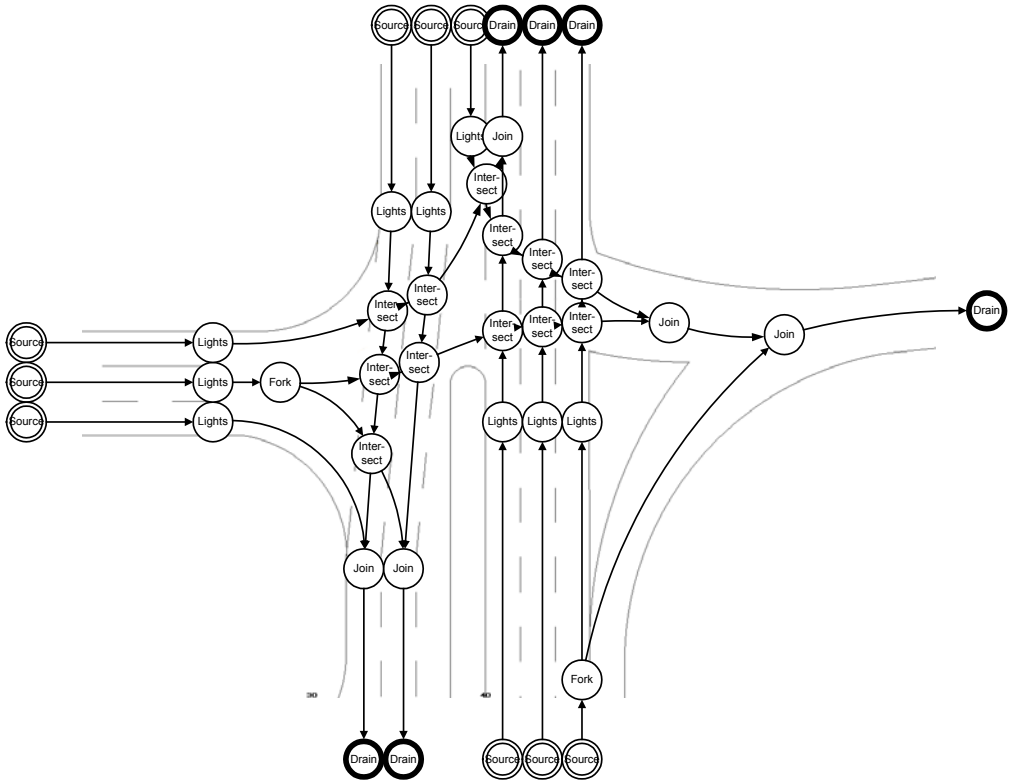


Figure 2: Nodes and tracks in the intersection, mapped on lanes

- Each track has a given *length*, a node where it origins from (*source node*), and a *destination node*. Tracks in curves have no radius, but a speed limed at a given position on the track (*velLimit* and *velLimitPos*) that is derived from the radius. Tracks and nodes are not decomposed any further.
- To each track a queue is assigned, managing the vehicles on the track. The length of each queue is passed on to the overall model level and provided as model output every 5 s of simulated time. The capacity of the queue is limited by the length of the track and the length of the vehicles on the track.

Tracks are connected by *nodes*. Figure 3 depicts the assignment of identifiers to the tracks and the association between tracks and nodes. Nodes do not have any spatial expansion.

- A source node reflects the point where vehicles enter the area of observation. The input parameter "arrival rate" is passed down form the overall model layer and considered in the source nodes for generation of vehicles.
- A lights node reflects the point where the first vehicle stops at red lights (white line on the road). Its states are *blocked* and *free*.
- A fork node reflects a point where drivers may choose between two directions.
- An intersect node reflects the intersection of two paths. Note, the two intersecting paths have different priority according to traffic regulations. Its states are *blocked* and *free*, which affects the lower prioritized lane through it.
- A join node reflects the point where two paths join. Note, the two joining paths have different priority according to traffic regulations. Its states are *blocked* and *free*, which affects the lower prioritized lane entering it.
- A drain node reflects the point where the vehicle leaves the area of observation.

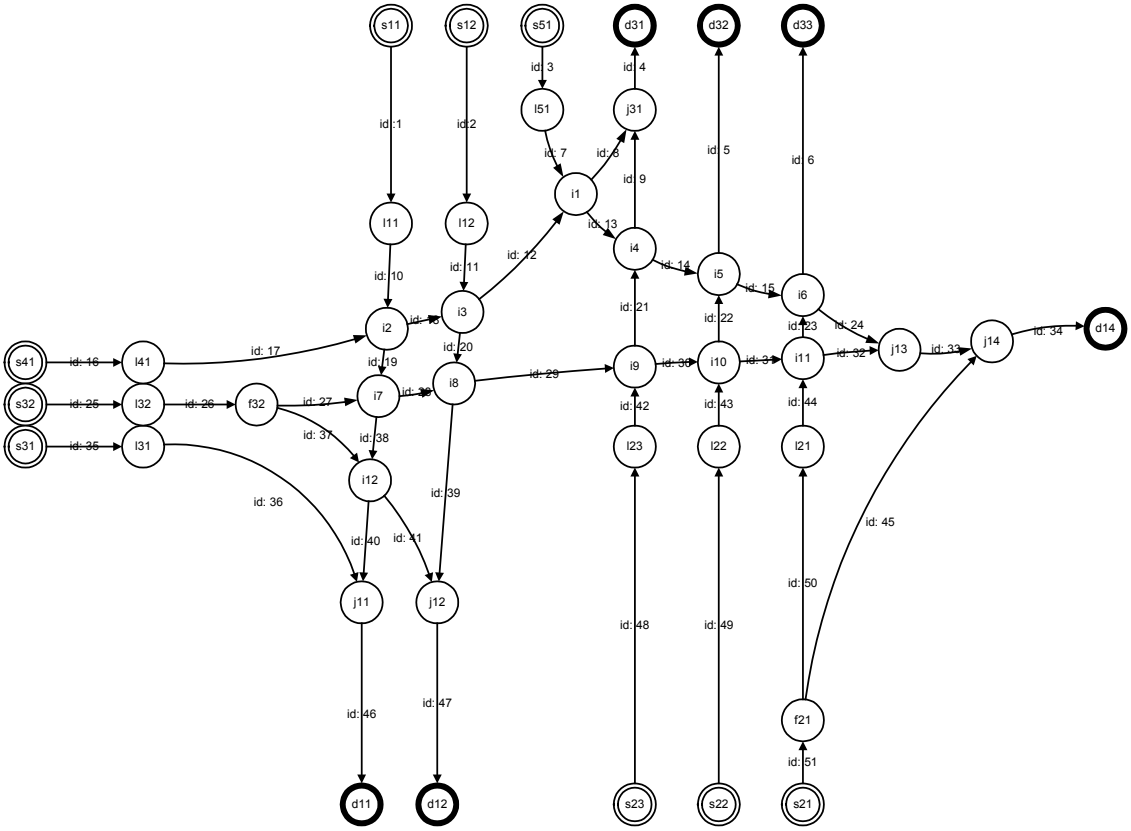


Figure 3: Paths and nodes with Ids

PATH	Length	velLimitPos	velLimit
p01	35,00	-1	-1
p02	35,00	-1	-1
p03	30,00	-1	-1
p04	28,75	-1	-1
p05	39,35	-1	-1
p06	40,61	-1	-1
p07	5,18	-1	-1
p08	6,42	-1	-1
p09	8,32	-1	-1
p10	6,08	-1	-1
p11	4,37	-1	-1
p12	6,08	3	7
p13	2,46	1	7
p14	3,40	-1	-1
p15	2,80	-1	-1
p16	30,00	-1	-1
p17	14,13	-1	-1
p18	3,16	-1	-1
p19	3,18	-1	-1
p20	3,64	-1	-1
p21	4,48	-1	-1
p22	2,05	-1	-1
p23	0,96	-1	-1
p24	5,78	-1	-1
p25	30,00	-1	-1
p26	1,25	-1	-1

PATH	Length	velLimitPos	velLimit
p27	12,23	-1	-1
p28	2,90	-1	-1
p29	5,98	-1	-1
p30	2,51	-1	-1
p31	2,51	-1	-1
p32	5,62	-1	-1
p33	16,42	-1	-1
p34	16,85	-1	-1
p35	30,00	-1	-1
p36	16,96	8	5
p37	12,75	6	5
p38	9,30	-1	-1
p39	17,11	-1	-1
p40	6,57	-1	-1
p41	6,88	-1	-1
p42	8,45	-1	-1
p43	8,59	-1	-1
p44	8,44	-1	-1
p45	72,71	50	7
p46	50,20	-1	-1
p47	50,23	-1	-1
p48	60,00	-1	-1
p49	60,00	-1	-1
p50	55,50	-1	-1
p51	4,50	-1	-1

Table 1: Tracks data

3.3.2.2 Behavior Description of Intersection Geometry

The intersection geometry is mainly passive, the only self-driven changes are initiated by the node "source". The structure of the intersection description is static and does not change.
The state of the intersection is defined by:

- the parameter values of the source nodes which are constant over a simulation run;
- the colors of its lights. They are changed by LightsCtrl;
- the positions of the vehicles. They are changed by the vehicles themselves;
- the states of the *intersect nodes* and the *join nodes*;
- the numbers of vehicles in the *drain nodes* and their accumulated *waiting times*.

The only self-driven sub-model in intersection geometry is the

- vehicle source: For initialization, the inter arrival time is read from the input of the overall model. As time advances, the vehicle sources "produce" vehicles following an exponential distribution. The inter-arrival rates preset defines, how frequently vehicles are generated in each source.

Behavioral constraint:

- The minimum inter arrival time must not be lower than 0.27 s – with a maximum speed of 15 m/s and a vehicle length of 4 m this would imply that the absolute distance between the rear buffer of the predecessor and the front buffer of the successor is negative.

Any other changes within the intersection are driven by *LightsCtrl* or *Vehicle*. *LightsCtrl* switches the state of node "lights" between *blocked* (red, redyellow, yellow) and *free* (green, black). Vehicles queue in the queues on the tracks, and leave the queues on their own again.

3.3.2.3 LightsCtrl Structure Description

Reflects the main control device of all lights in the complete intersection. Changes the *lights colors* [red, redyellow, green, yellow, black] over time, according to the lights switching table read during initiation. LightsCtrl is intended to model the I/O behavior of the main lights control device. Its output attributes are a vector of color codes (one color for each group of lights in the intersection), indicating the color codes of all lights within the intersection. LightsCtrl is not decomposed any further.

LighsCtrl	Lights color: FV01
	Lights color: FV02
	Lights color: FV03
	Lights color: FV04
	Lights color: DN05
	Switching Table [NrStates] [Lights colors, duration]

3.3.2.4 LightsCtrl Behavior Description

The switching tables reprinted in Appendix C.1 document the switching patterns for the control of the real lights in the intersection. The values of the output parameters of lightsCtrl are intended to exactly reflect the behavior specified for the real lights. Lights Control is completely self-driven, uninterruptible, and updates its outputs FV01 – FV04 and DN05 after the expiration of the duration of each state. The switching information is read from the configuration file in the beginning of each run. The state of lightsCtrl is defined by the current colors of the lights (lights color vector). The state transitions are triggered by the expiration of each state's duration.

Behavioral constraints:

- A lights cycle (FVxx) must always be red – redyellow – green – yellow – red.

3.3.2.5 Vehicle Structure Description

The sub-model *vehicle* reflects a single vehicle. It moves through the intersection with a given speed, decelerates and accelerates as appropriate, occupies space, or blocks the road. Each vehicle has a length, indicating the space it occupies (front bumper to rear bumper). The position of a vehicle is defined by the track it is on and the position on the track. The vehicle structure is not decomposed any further.

Vehicle	Length = 4m
	Desired Velocity = 15 m/s
	Preferred Acceleration = 3 m/s^2
	Preferred Deceleration = -4 m/s^2
	Current Velocity $\in$ [0; 15] m/s
	Current Ac-/Deceleration $\in$ {]-∞; 0], 3} m/s^2
	Current Path $\in$ {1, 2, …, 51}
	Current Position $\in$ [0; pathLength]

Length, Desired Velocity, Preferred Ac- / Deceleration are equal for all Vehicles.

- The length was taken from the technical data sheet of the VW Golf III.
- The desired velocity of 15 m/s (=54 km/h) is assumed to be representative for the cruising speed in a German city with speed limit of 50 km/h.
- Preferred acceleration and deceleration are guesses. The acceleration of 3 m/s^2 is based on the assumption that speedy acceleration can easily accelerate a car from standing to 100 km/h (ca. 27.78 m/s) in 10 seconds. The preferred deceleration is a pure guess.

The slight deviations in these values are not considered to be relevant for the examination of queuing behavior. The vehicle is not decomposed any further. The vehicle records its time in the intersection and its waiting time. Also it provides these times as output on leaving the intersection.

3.3.2.6 Vehicle Behavior Description

When the vehicle is instantiated and enters the modeled cutout of reality, it is assumed to travel with its *desired velocity* v_{des} or the *allowed velocity* v_{allow}, whichever is lower. The *position* x_{i+1} of the vehicle after *time* Δt is updated according to its previous *position* x_i and its *current velocity* v_i. The velocity is updated according to the vehicle's *acceleration* a and the *target velocity* v_{tar}.

$$v_{i+1} = v_i + a\Delta t \tag{1}$$

If $v_{i+1} \leq v_{tar}$ (for $a > 0$ and $v_i \leq v_{tar}$), or $v_{i+1} \geq v_{tar}$ (for $a < 0$ and $v_i \geq v_{tar}$), x_{i+1} is determined directly:

$$x_{i+1} = x_i + v_i\Delta t + \frac{1}{2}a\Delta t^2 \tag{2}$$

Otherwise, the fraction Δt_{change} of Δt is determined, after which v_{tar} is reached.

$$\Delta t_{change} = \frac{v_{tar} - v_i}{a} \tag{3}$$

Then $v_{i+1} = v_{tar}$, and

$$x_{i+1} = x_i + v_i \Delta t_{change} + \frac{1}{2} a \Delta t_{change}^2 + v_{tar} \cdot \left(\Delta t - \Delta t_{change} \right) \tag{4}$$

Afterwards, a is set to 0.

The vehicle checks frequently for obstacles in its path and determines the required deceleration until the next check. The *allowed velocity* depends on the distance to the next obstacle. If there is no obstacle ahead, the vehicle accelerates with the *preferred acceleration* $a_{pospref}$ to its *desired velocity*. The following reasons cause the vehicle to change its *acceleration* below zero (deceleration).

- Red lights ahead: When approaching the lights, the necessity of deceleration is calculated as a function of the own current velocity v_i and the remaining distance to the lights $x_{lights} - x_i$.

$$a = \frac{-v_i^2}{2\left(x_{lights} - x_i\right)} \tag{5}$$

  If the required deceleration a is lower then the preferred deceleration $a_{negpref}$ (which means, there is a need to decelerate), or if

$$x_{lights} - x_i \le v_i \Delta t + \frac{1}{2} a_{pospref} \Delta t^2 \tag{6}$$

  (which means that accelerating as preferred carries the vehicle too far), a is considered to be the constant deceleration that makes the vehicle stop exactly at the lights to achieve $v_{tar} = 0$. Otherwise, the red lights are not (yet) considered as an obstacle.

- Narrow curve ahead: The necessity of deceleration when approaching a narrow curve is calculated as a function of the current own velocity v_i, the remaining distance to the x_{Limit} of the curve (velLimitPos), and the speed v_{Limit} that is recommended in the curve (velLimit), compared to the assumed *preferred deceleration*.

$$a = \frac{v_i \left(v_{Limit} - v_i\right) + \frac{1}{2}\left(v_{Limit} - v_i\right)^2}{x_{Limit} - x_i} = \frac{v_{Limit}^2 - v_i^2}{2\left(x_{Limit} - x_i\right)} \tag{7}$$

  If the required deceleration a is lower then the preferred one $a_{negpref}$ (which means there is a need to decelerate), or

$$x_{Limit} - x_i \le v_i t + \frac{1}{2} a_{pospref} t^2 \,. \tag{8}$$

a is considered to be the constant deceleration that makes the vehicle pass the velLimitPos x_{Limit} with exactly the target velocity $v_{tar} = v_{Limit}$.

- Slower vehicle ahead: It is assumed that each vehicle holds a safety distance to its predecessor that depends on its own speed and its predecessor's speed. Idealizing the complex process of safety distance determination and regulation, we assume that there is a velocity limit for each vehicle, which equals the velocity of its predecessor. The position of this velocity limit moves behind the predecessor, following it in the safety distance.
  First the estimated position $x_{predestim}$ and estimated velocity $v_{predestim}$ of the predecessor after Δt are determined. The safety distance x_{safe} is calculated as $\min(v_i, v_{predestim}) + 1\text{m} + x_{size}$. (This is a rough translation of a rule of thumb for safety distance determination, recommended by the German AAA.) Then the required deceleration a is calculated:

$$a = \frac{v_{predestim}^2 - v_i^2}{2\left(x_{predestim} - x_{safe} - x_i\right)} \tag{9}$$

  If the required deceleration a is lower then the preferred deceleration $a_{negpref}$ (which means there is a need to decelerate), or

$$x_{predestim} - x_{safe} - x_i \le v_i \Delta t + \frac{1}{2} a_{pospref} \Delta t^2 \tag{10}$$

  a is considered to be the constant deceleration that makes the vehicle pass a "follow point" $x_{predestim} - x_{safe}$ with exactly the target velocity $v_{tar} = v_{predestim}$.

- "Blocked" intersect ahead: The vehicle must yield to vehicles on lanes with higher priority, if lanes intersect. If an intersect node is blocked and the velocity is higher than the *allowed velocity*, a constant deceleration is calculated that fully stops the vehicle at the intersect node, if necessary. It accelerates again, when the intersect node becomes unblocked.

- "Blocked" join ahead: The vehicle must yield to vehicles on lanes with higher priority, if lanes intersect. If a join node is blocked, the vehicle fully stops at the intersect node, if necessary. It accelerates again, when the join node becomes unblocked.

If several conditions apply, the required deceleration for all of them is calculated separately, and the lowest is chosen.

Other actions:

- "Block" and "free" intersect or join node: If a vehicle closes in to an intersect or join node on the higher prioritized path, it "blocks" the node for all vehicles approaching on the lower prioritized path, if it passes it during the next three seconds, constant velocity assumed. The node becomes unblocked by the vehicle again, if it is passed, or the vehicle's velocity decreases, until the 3s blocking condition becomes unvalid.

Behavioral constraints:

- FIFO / No overtaking: On a single path, no overtaking is allowed. If a vehicle overtakes its predecessor, there is an error in the model.

3.3.3 Object Inheritance Hierarchies

Source, lights, fork, intersect, drain, and join are nodes. No other inheritance is defined.

node	Source
	Lights
	Fork
	Intersect
	Join
	Drain

3.3.4 Integrated Predefined Submodels

No predefined sub-models were integrated.

3.4 Evaluated Information of the Real System

- Lights switching diagrams, Appendix C.1
- CAD drawing of intersection, scale 1:500, Appendix C.2
- Induction loops data, Appendix C.3
- Technical data sheet of VW Golf III (length)
- Visual, subjective perception of intersection layout

4 FORMAL MODEL

(Documented according to the Formal Model template in Appendix A of the main document.)

4.1 Configuration Control Information

Version Identifier: ADP-FM-V1, based on ADP-CM-V1

4.2 Identification of the Formalism

The complete Formal Model was specified using the DEVS formalism, introduced in [Zeigler, Praehofer, and Kim 2000]

4.3 Structure and Behavior Description

ColorCodes = {„red“, „yellow“, „green“, „redyellow“, “black”}

|Intersects| = 12;
|Joins| = 5;
|Lights| = 9;
|lightsGroups| = 5;
|lightsStates| = 14;
|Tracks| = 51;
|Sources| = 9;
|Drains| = 6;
|Queues| = |Tracks| + |Sources| + |Drains|

LightsStatesTab = $\{(a_{i,j})_{i = 1,...,|lightsStates|,\ j = 1,...,|lightsGroups| + 2}$:
// id in 1st column
$a_{i,j} \in \{1, 2, ..., |\text{lightsStates}|\}$ for $i = 1,..., |\text{lightsStates}|, j = 1$
// duration in 2nd column
$a_{i,j} \in R_0^+$ for $i = 1,..., |\text{lightsStates}|, j = 2$
// color codes in 3rd to last column
$a_{i,j} \in$ ColorCodes
for $i = 1,..., |\text{lightsStates}|, j = 3,..., |\text{lightsGroups}| + 2$
}

4.3.1 LightsCtrl (Layer 1 Submodel)

LightsCtrl is defined in Parallel DEVS as

$$DEVS_{lightsCtrl} = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta),$$

where

InPorts = {„LightsStatesTab“}, where
$X_{LightsStatesTab}$ = LightStatesTab

$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values;

OutPorts = {„LightsVec“}, where
$Y_{LightsVec} = \text{ColorCodes}^{|\text{LightsGroups}|}$

$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values;

// State is
current phase, phase duration $\sigma \in R_0^+$, current lightStatesTable $L \in A$

$S = \{$„initial“, „prepLightsState(1)“, ... , „prepLightsState(|lightsStates|)“$\} \times R_0^+ \times A$

δ_{ext} (phase, σ, L, e, („LightsStatesTab“, x))
= („prepLightsState1“, 0, x) if phase = „initial“
= (phase, σ - e, L) otherwise

δ_{int} (phase, σ, L)
= („prepLightsState2“, $l_{1,2}$, L) if phase = „prepLightsState(1)“
= („prepLightsState3“, $l_{2,2}$, L) if phase = „prepLightsState(2)“
...
= („prepLightsState(|lightsStates|)“, $l_{13,2}$, L) if phase = „pre-pLightsState(|lightsStates| - 1)“
= („prepLightsState1“, $l_{|lightsStates|,2}$, L) if phase = „pre-pLightsState(|lightsStates|)“

δ_{con} (s, ta(s), x) = δ_{ext} (δ_{int} (s), 0, x)

λ (phase, σ, L)
= (LightsVec, $(l_{1,3}, \ldots, l_{1,|lightsGroups|})^T$) if phase = prepLightsState(1)
= (LightsVec, $(l_{2,3}, \ldots, l_{2,|lightsGroups|})^T$) if phase = prepLightsState(2)
...
= (LightsVec, $(l_{|lightsStates|,3}, \ldots, l_{|lightsStates|,|lightsGroups|})^T$) if phase = pre-pLightsState(|lightsStates|)

ta (phase, σ, L)
= ∞ if phase = “initial”
= σ otherwise

4.3.2 IntersGeom (Layer 1 Submodel)

IntersGeom is defined in Parallel DEVS as

$DEVS_{IntersGeom} = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta)$,

where
InPorts = {“LightsVec”, “VehicleVec”}, where
$X_{LightsVec}$ = ColorCodes$^{|LightsGroups|}$, where
$X_{VehicleVec} = V^n$ where
n is the number of vehicles and
$V = N \times N \times R_0^+ \times R_0^+ \times R$

$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values;

OutPorts = {„NodesStatesVec“, “QLengthVec”}, where
$Y_{NodesStatesVec}$ = {“free”, “blocked”}$^{|Intersects| + |Joins| + |Lights|}$

$Y_{QLengthVec} = N_0^{|Queues|}$

$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values;

// State is defined by
phase ∈ {passive, update},
time σ to ql update ∈ R_0^+,
qLengthsVec ∈ $N_0^{|Queues|}$,
lightsStates ∈ {„free“, „blocked“}$^{|Lights|}$,
joinsStates ∈ {„free“, „blocked“}$^{|Joins|}$,
intersectsStates ∈ {„free“, „blocked“}$^{|Intersects|}$,

S = {„passive“, „update“} × R_0^+ × $N_0^{|Queues|}$ × {„free“, „blocked“}$^{|Intersects|}$ × {„free“, „blocked“}$^{|Joins|}$ × {„free“, „blocked“}$^{|Lights|}$

δ_{ext} (phase, σ, queueSV, lightsSV, joinsSV, intersectsSV, e, („LightsVec“, x))
= („update“, σ - e, queueSV, getNewLightsSV(x), joinsSV, intersectsSV)

where
getNewLightsSV(x) :
ColorCodes$^{|LightsGroups|}$ → {„free“, „blocked“}$^{|Lights|}$

δ_{ext} (phase, σ, queueSV, lightsSV, joinsSV, intersectsSV, e, („VehicleVec“, x))
= („update“, σ - e, getNewQueueSV(x), lightsSV, getNewJoinsSV(x), get-NewIntersectsSV(x))

where
getNewQueueSV(x) : $V^n \rightarrow N_0^{|Queues|}$
getNewJoinsSV(x) : $V^n \rightarrow$ {„free“, „blocked“}$^{|Joins|}$
getNewIntersectsSV(x) : $V^n \rightarrow$ {„free“, „blocked“}$^{|Intersects|}$

// report
δ_{int} („passive”, σ, queueSV, lightsSV, joinsSV, intersectsSV)
= („passive”, reportIntervall, queueSV, lightsSV, joinsSV, intersectsSV)

// return to report state after update
δ_{int} („update”, σ, queueSV, lightsSV, joinsSV, intersectsSV)
= („passive”, σ, queueSV, lightsSV, joinsSV, intersectsSV)

δ_{con} = (s, ta(s), x) = δ_{ext} (δ_{int} (s), 0, x)

λ („passive”, σ, queueSV, lightsSV, joinsSV, intersectsSV)
= („QLengthVec”, queueSV)

λ („update”, σ, queueSV, lightsSV, joinsSV, intersectsSV)
= („NodesStatesVec”, (lightsSV, joinsSV, intersectsSV))

ta (phase, σ, queueSV, lightsSV, joinsSV, intersectsSV)
= σ if phase = “passive”
= 0 if phase = “update”;

4.3.3 VehicleGen (Layer 2 Submodel "Vehicle Source")

VehicleGen is defined in Parallel DEVS as

$$DEVS_{VehicleGen} = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta),$$

where

InPorts = {„InterArrTVec"}, where
$X_{InterArrTVec} = (R_0^+)^{|Gen|}$, |Gen| = 9

$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values;

OutPorts = {„NewVehicle"}, where
$Y_{NewVehicle}$ = V where $V = N \times N \times R_0^+ \times R_0^+ \times R$

$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values;

// State is
phase ∈ {"initial", "generating"}, gen interval $\sigma \in R_0^+$, interArrT $\in R_0^+$

S = {"initial", "generating"} $\times R_0^+ \times R_0^+$

δ_{ext} (phase, σ, interArrT, e, („InterArrTVec", x))
= („generating", 0, getOwnInterArrT(x)) if phase = „initial"
= (phase, σ - e, interArrT) otherwise

δ_{int} ("generating", σ, interArrT)
= („generating", exp(interArrT), interArrT)

δ_{con} (s, ta(s), x) = δ_{ext} (δ_{int} (s), 0, x)

λ ("generating", σ, interArrT)
= ("VehicleData", getVehicleData())

ta (phase, σ, L)
= σ if phase = "generating"
= ∞ otherwise

4.3.4 VehicleCtrl (Layer 1 Submodel)

VehicleCtrl is defined in Parallel DEVS as

$$DEVS_{vehicleCtrl} = (X_M, Y_M, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta),$$

where

InPorts = {„nodesStatesVec", "NewVehicle"}, where

$X_{NodesStatesVec}$ = {„free“, „blocked“}$^{|Intersects|}$ × {„free“, „blocked“}$^{|Joins|}$ ×{„free“, „blocked“}$^{|Lights|}$

$X_{NewVehicle} = V$, where $V = N \times N \times R_0^+ \times R_0^+ \times R$

$X_M = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values;

OutPorts = {„TWait“, “VehicleVec” }, where

$Y_{TWait} = R_0^+$

$Y_{VehicleVec} = V^n$ where n is number of vehicles

$Y_M = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values;

// State is

phase ∈ {“updating”},

time-step σ ∈ R_0^+,

vehicleVec ∈ V^n,

mapState ∈ {„free“, „blocked“}$^{|Intersects|}$ × {„free“, „blocked“}$^{|Joins|}$ ×{„free“, „blocked“}$^{|Lights|}$

S = {“updating”} × R_0^+ × V^n × {„free“, „blocked“}$^{|Intersects|}$ × {„free“, „blocked“}$^{|Joins|}$ ×{„free“, „blocked“}$^{|Lights|}$

δ_{ext} (“updating”, σ, vehicleVec, mapState, e, („NewVehicle“, x))

= („updating“, σ - e, addVehicle(x), mapState)

δ_{ext} (“updating”, σ, vehicleVec, mapState, e, („NodesStatesVec“, x))

= (“updating”, σ - e, vehicleVec, x)

δ_{int} (“updating”, σ, vehicleVec, mapState)

= (“updating”, TIME_STEP, update(vehicleVec), mapState)

where update(vehicleVec) : $V^n \rightarrow V^n$

δ_{con} (s, ta(s), x) = δ_{ext} (δ_{int} (s), 0, x)

λ (“updating”, σ, vehicleVec, mapState)

= (“VehicleVec”, vehicleVec)

= (“TWaitVec”, tWaitofArrivedVehicles(vehicleVec))

ta (phase, σ, L) = σ

4.3.5 Intersection (Overall Integrated Layer 0 Model)

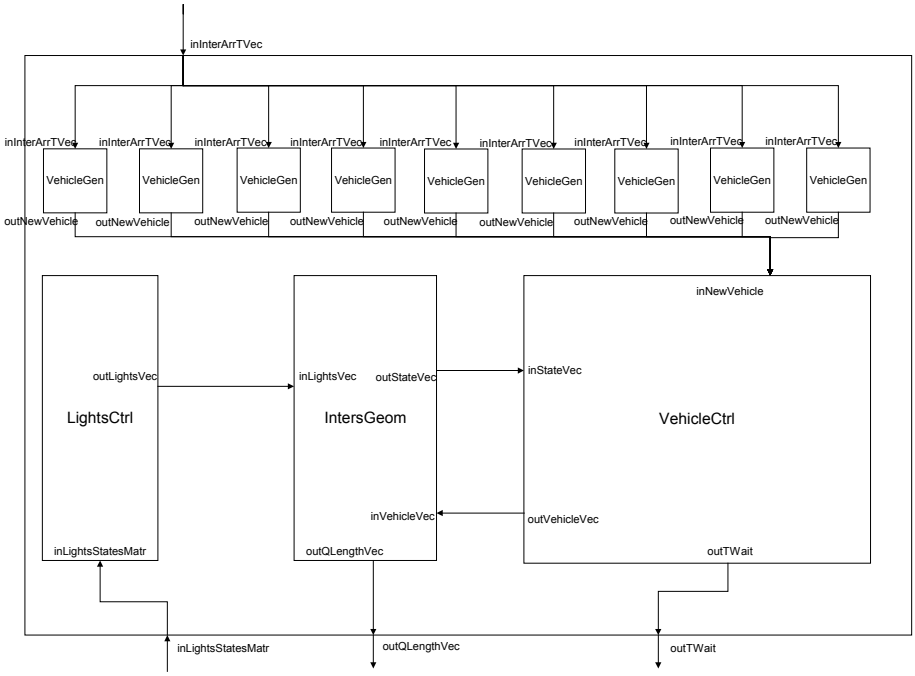


Figure 4: Sketch of the Coupled Model "Intersection"

The coupled DEVS specification of Intersection is

$$N = (X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC),$$

where

InPorts = {„InterArrTVec“, “LightsStatesTab”}, where

$X_{InterArrTVec} = (R_0^+)^{|Gen|}$, |Gen| = 9

$X_{LightsStatesTab}$ = LightsStatesTab

$X = \{(p, v) \mid p \in InPorts, v \in X_p\}$ is the set of input ports and values;

OutPorts = {„TWait“, “QLengthVec” }, where

$Y_{TWait} = R_0^+$

$Y_{QLengthVec} = N_0^{|Queues|}$

$Y = \{(p, v) \mid p \in OutPorts, v \in Y_p\}$ is the set of output ports and values;

D = {LightsCtrl, IntersGeom, VehicleCtrl, VehicleGen1, VehicleGen2,…, VehicleGen9}

$M_{VehicleGen1} = M_{VehicleGen2} = \ldots = M_{VehicleGen9} = M_{VehicleGen}$

EIC = {((N, „InterArrTVec“), (VehicleGen1, „InterArrTVec“),(VehicleGen2, „InterArrTVec“), …, (VehicleGen9, „InterArrTVec“)), ((N, “LightsStatesTab”), (LightsCtrl, “LightsStatesTab”))}

EOC = {((IntersGeom, “QLengthVec”), (N, “QLengthVec”)), ((VehicleCtrl, „TWait“), (N, „TWait“))}

IC = {((VehicleGen1, “NewVehicle”), (VehicleGen2, “NewVehicle”), …, (VehicleGen9, “NewVehicle”), (VehicleCtrl, “NewVehicle”)), ((LightsCtrl, “LightsVec”), (IntersGeom, “LightsVec”)), ((intersGeom, “NodesStatesVec”), (VehicleCtrl, “nodesStatesVec”)), ((VehicleCtrl, “VehicleVec”), (IntersGeom, “VehicleVec”))}

5 EXECUTABLE MODEL

(Documented according to the Executable Model template in Appendix A of the main document.)

5.1 Configuration Control Information

Version Identifier: ADP-EM-V1, based on ADP-FM-V1

5.2 Runtime Environment

MS Windows 2000 PC, SLX Professional [Henriksen 1998]. The model behavior is visualized with Proof Animation [Wolverine 2003].

5.3 SW Description

All input to the model is provided by the file "ExpFile.txt".
Output is recorded in the files

- "vehicleTimes.txt" : Each vehicle inserts its alive time and its waiting time.
- "IntersStateTraj.txt" : Whenever the state of the layer 1 submodel intersection geometry changes, this is recorded here.
- "AnDerPointAnimV2.atf" : This is the simulation trace file required for the visualization of model behavior.

The SLX code files are attached on the CD-ROM.

5.4 Simulation Infrastructure

The simulation infrastructure provided by SLX was used. More information about it can be obtained at [Wolverine 2003]

5.5 Standards Summary

No standards were applied for the implementation of the model.

6 SIMULATION RESULTS

6.1 Configuration Control Information

Version Identifier: ADP-SR-V1, based on ADP-EM-V1

6.2 Execution Environment

An excerpt of the experiment setup file (ExpFile) is documented in Appendix C.4. The model output was written into .txt file, and post-processed with MS Excel.

6.3 Experimental Design

6.3.1 Aim of Experimentation

The performance of the intersection is observed, using three different lights control strategies and different intensities of traffic. The accumulated waiting times of the vehicles are supposed to contribute to the identification of the most appropriate lights switching pattern.

6.3.2 Experiment Setup

6.3.2.1 Density of Traffic Flow

There are three intersection entries, which will be distinguished during experimentation:

- North: Sources s11, s12, s51
- South: Sources s21, s22, s23
- West: Sources s31, s32, s41

The density of traffic on one entry road is assumed to be equally distributed and parameterized for all three lanes.

The density of traffic is discretized into three categories:

- Low traffic density: The inter arrival time between vehicles is 60 seconds
- Medium traffic density: The inter arrival time between vehicles is 30 seconds
- High traffic density: The inter arrival time between vehicles is 10 seconds

The inter arrival time for low traffic was derived from the induction sensors data documented in the associated files.

6.3.2.2 Applied lights switching patterns:

The following lights switching patterns are distinguished:

Id	Duration [s]	FV01	FV02	FV03	FV04	DN05
1	25	green	green	red	red	black
2	3	yellow	green	red	red	black
3	1	red	green	red	red	black
4	1	red	green	redyellow	red	black
5	3	red	yellow	green	red	black
6	2	red	red	green	red	black
7	1	red	red	green	red	green
8	1	red	red	green	redyellow	green
9	20	red	red	green	green	black
10	3	red	red	yellow	yellow	black
11	2	red	red	red	red	black

Id	Duration [s]	FV01	FV02	FV03	FV04	DN05
12	1	red	redyellow	red	red	black
13	1	redyellow	green	red	red	black
14	6	green	green	red	red	black

Table 2: Switching Pattern 1

The differentiation between *state 1* and *state 14* is due to the necessity to have a green period of 25 seconds after initiation of this switching program.

Id	Duration [s]	FV01	FV02	FV03	FV04	DN05
1	20	green	green	red	red	black
2	3	yellow	green	red	red	black
3	2	red	green	redyellow	red	black
4	3	red	yellow	green	red	black
5	2	red	red	green	red	black
6	1	red	red	green	red	green
7	1	red	red	green	redyellow	green
8	28	red	red	green	green	black
9	3	red	red	yellow	yellow	black
10	2	red	red	red	red	black
11	1	red	red yellow	red	red	black
12	1	red yellow	green	red	red	black
13	22	green	green	red	red	black

Table 3: Switching Pattern 2

Id	Duration [s]	FV01	FV02	FV03	FV04	DN05
1	15	green	green	red	red	black
2	3	redyellow	green	red	red	black
3	1	red	green	red	red	black
4	1	red	green	redyellow	red	black
5	17	red	green	green	red	black
6	3	red	redyellow	green	red	black
7	2	red	red	green	red	black
8	1	red	red	green	red	green
9	1	red	red	green	redyellow	green
10	10	red	red	green	green	black
11	3	red	red	redyellow	redyellow	black
12	2	red	red	red	red	black
13	1	red	redyellow	red	red	black
14	1	redyellow	green	red	red	black
15	28	green	green	red	red	black

Table 4: Switching Pattern 3

6.3.2.3 Experiment Duration

Each single simulation run lasts 3600 seconds (1 hour) of simulated time. This is assumed to include a sufficient number of stochastic vehicle generations to grant statistical relevance.

6.3.2.4 Experiment Overview

For each lights switching table, simulation runs with traffic density parameters documented in Table 5 are executed, yielding a total of 81 simulation runs in the experiment.

6.4 Input Data Documentation

Table 5 documents the initialization values of the model input parameters for each simulation run of the experiment.

Run ID	Density N	Density S	Density W	Density N (s)	Density S (s)	Density W (s)
1	L	L	L	60	60	60
2	M	L	L	30	60	60
3	L	L	M	60	60	30
4	L	M	L	60	30	60
5	M	L	M	30	60	30
6	M	M	L	30	30	60
7	L	M	M	60	30	30
8	M	M	M	30	30	30
9	H	L	L	10	60	60
10	L	L	H	60	60	10
11	L	H	L	60	10	60
12	H	L	H	10	60	10
13	H	H	L	10	10	60
14	L	H	H	60	10	10
15	H	H	H	10	10	10
16	H	M	M	10	30	30
17	M	M	H	30	30	10
18	M	H	M	30	10	30
19	H	M	H	10	30	10
20	H	H	M	10	10	30
21	M	H	H	30	10	10
22	L	H	M	60	10	30
23	L	M	H	60	30	10
24	M	H	L	30	10	60
25	M	L	H	30	60	10
26	H	M	L	10	30	60
27	H	L	M	10	60	30

Table 5: Combinations of incoming traffic densities

6.5 Results – Excerpts

Microsoft Excel - SimResAusw.xls

Datei Bearbeiten Ansicht Einfügen Format Extras Daten Fenster ? Acrobat

L2935 =

	A	B	C	D	E	F	G	H	I
1	Vehicle	11-1:	Drain	11	tAlive:	750	tWait:	0	
2	Vehicle	11-10:	Drain	11	tAlive:	2750	tWait:	1700	
368	Vehicle	11-97:	Drain	11	tAlive:	1500	tWait:	450	
369	Vehicle	11-98:	Drain	11	tAlive:	1000	tWait:	0	
370	Vehicle	11-99:	Drain	11	tAlive:	725	tWait:	0	
371					**tAliveSumme:**	**715300,00**	**tWaitSumme:**	**373100**	
372					**tAliveMittel:**	**1933,24**	**tWaitMittel:**	**1008,38**	
373									
374									
375	Vehicle	12-1:	Drain	12	tAlive:	750,00	tWait:	0	
376	Vehicle	12-10:	Drain	12	tAlive:	4475,00	tWait:	3425	
743	Vehicle	12-97:	Drain	12	tAlive:	3250,00	tWait:	2125	
744	Vehicle	12-98:	Drain	12	tAlive:	1550,00	tWait:	450	
745	Vehicle	12-99:	Drain	12	tAlive:	1550,00	tWait:	400	
746					**tAliveSumme:**	**766900,00**	**tWaitSumme:**	**421700**	
747					**tAliveMittel:**	**2067,12**	**tWaitMittel:**	**1136,66**	
748									
749									
750	Vehicle	13-1:	Drain	14	tAlive:	1275,00	tWait:	375	
751	Vehicle	13-10:	Drain	14	tAlive:	3325,00	tWait:	1925	
1108	Vehicle	13-97:	Drain	14	tAlive:	2100,00	tWait:	700	
1109	Vehicle	13-98:	Drain	14	tAlive:	2075,00	tWait:	675	
1110	Vehicle	13-99:	Drain	14	tAlive:	2075,00	tWait:	625	
1111					**tAliveSumme:**	**909275,00**	**tWaitSumme:**	**520175**	
1112					**tAliveMittel:**	**2518,77**	**tWaitMittel:**	**1440,93**	
1113									

Run3HHH | Run2HHH | Run1MMM | Roh | Run1LLL | Diagramm1 | **Run1H**

Bereit NF

Figure 5: Excerpt of the post-processed simulation output data

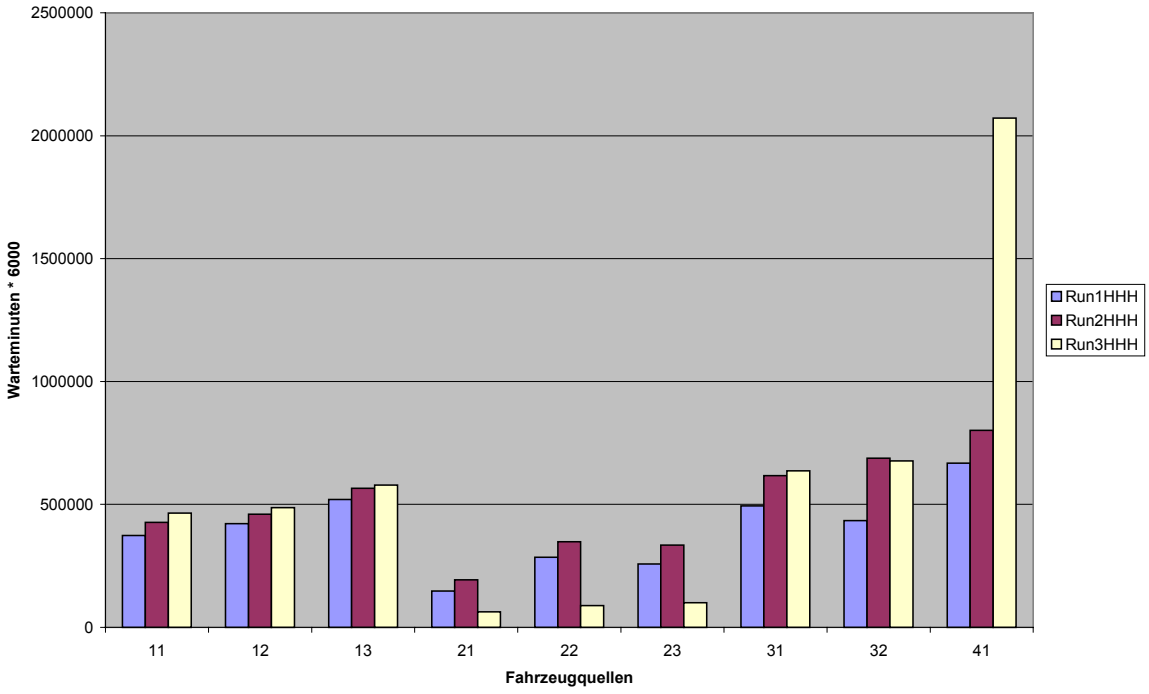
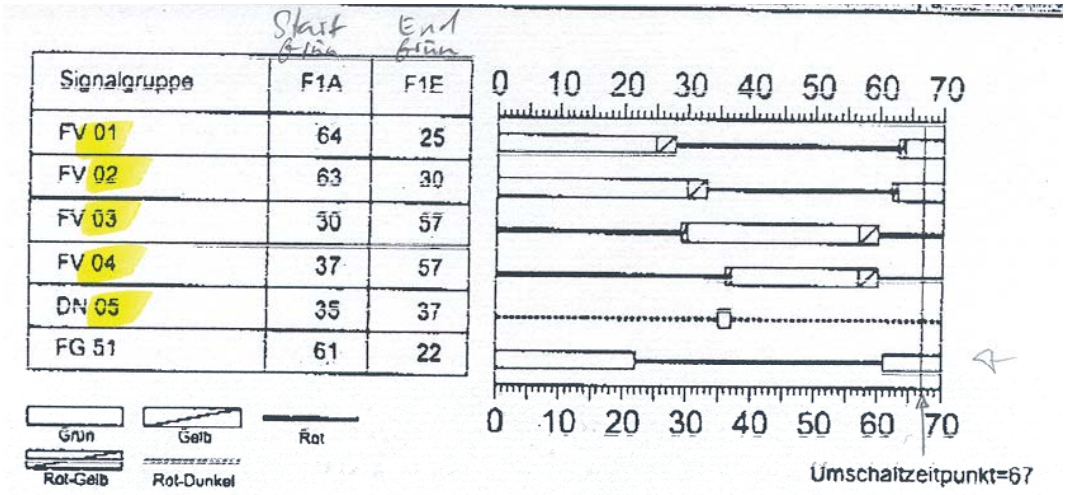


Figure 6: Comparison of the three different lights control strategies under heavy traffic from all directions

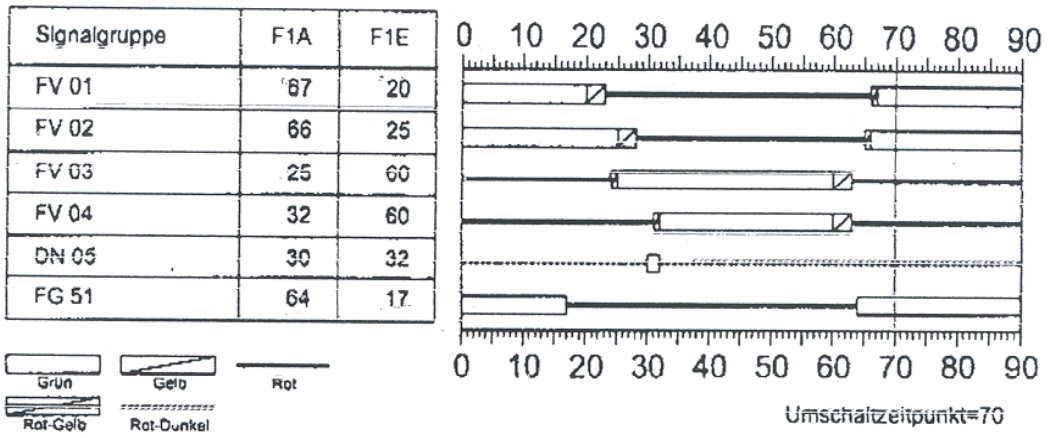
APPENDIX C.1 – LIGHTS SWITCHING TABLES

Table 1



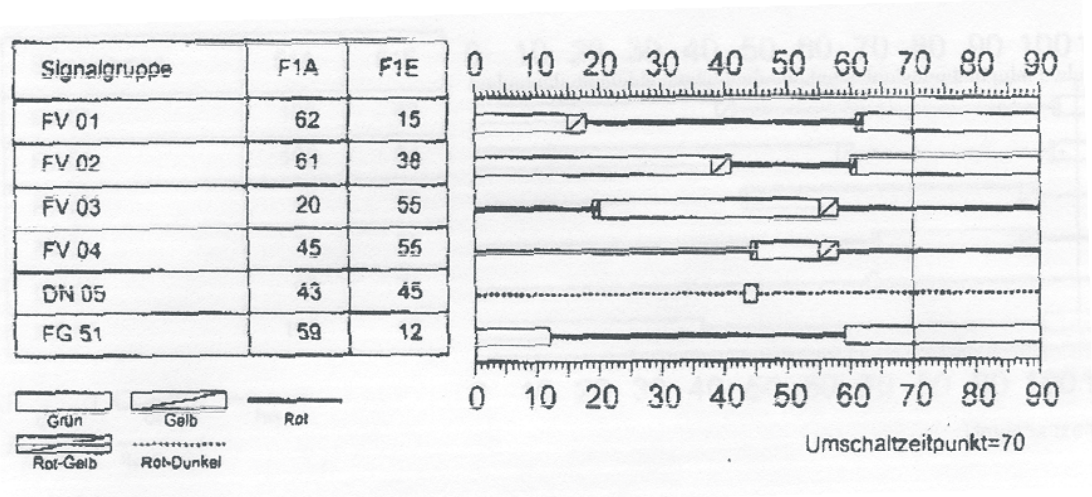
Signalgruppe	F1A	F1E
FV 01	64	25
FV 02	63	30
FV 03	30	57
FV 04	37	57
DN 05	35	37
FG 51	61	22

Table 2



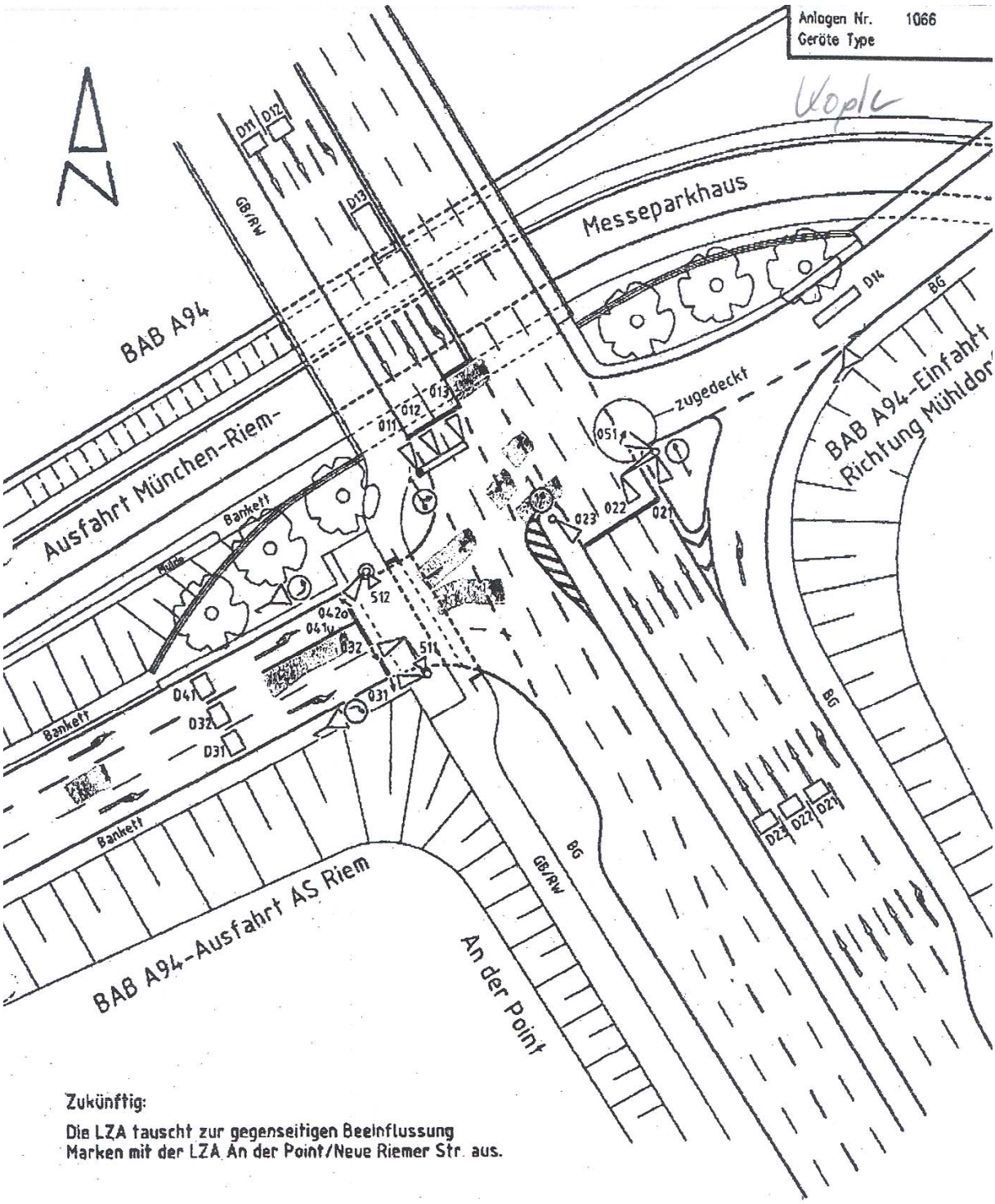
Signalgruppe	F1A	F1E
FV 01	67	20
FV 02	66	25
FV 03	25	60
FV 04	32	60
DN 05	30	32
FG 51	64	17

Table 3

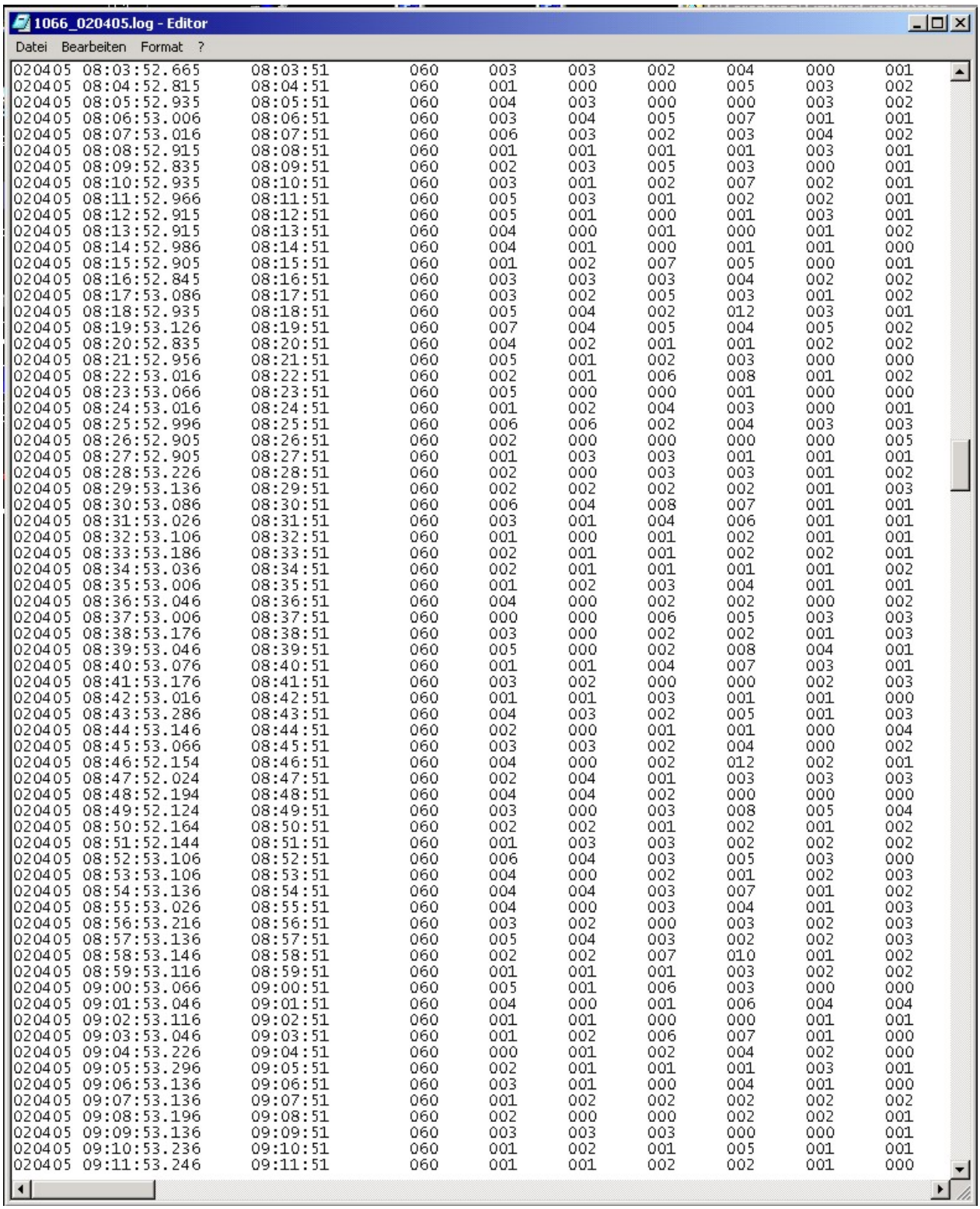


Signalgruppe	F1A	F1E
FV 01	62	15
FV 02	61	38
FV 03	20	55
FV 04	45	55
DN 05	43	45
FG 51	59	12

APPENDIX C.2 – CAD SKETCH



APPENDIX C.3 – EXEMPLARY INDUCTION SENSOR DATA



1066_020405.log - Editor

Datei Bearbeiten Format ?

020405	08:03:52.665	08:03:51	060	003	003	002	004	000	001
020405	08:04:52.815	08:04:51	060	001	000	000	005	003	002
020405	08:05:52.935	08:05:51	060	004	003	000	000	003	002
020405	08:06:53.006	08:06:51	060	003	004	005	007	001	001
020405	08:07:53.016	08:07:51	060	006	003	002	003	004	002
020405	08:08:52.915	08:08:51	060	001	001	001	001	003	001
020405	08:09:52.835	08:09:51	060	002	003	005	003	000	001
020405	08:10:52.935	08:10:51	060	003	001	002	007	002	001
020405	08:11:52.966	08:11:51	060	005	003	001	002	002	001
020405	08:12:52.915	08:12:51	060	005	001	000	001	003	001
020405	08:13:52.915	08:13:51	060	004	000	001	000	001	002
020405	08:14:52.986	08:14:51	060	004	001	000	001	001	000
020405	08:15:52.905	08:15:51	060	001	002	007	005	000	001
020405	08:16:52.845	08:16:51	060	003	003	003	004	002	002
020405	08:17:53.086	08:17:51	060	003	002	005	003	001	002
020405	08:18:52.935	08:18:51	060	005	004	002	012	003	001
020405	08:19:53.126	08:19:51	060	007	004	005	004	005	002
020405	08:20:52.835	08:20:51	060	004	002	001	001	002	002
020405	08:21:52.956	08:21:51	060	005	001	002	003	000	000
020405	08:22:53.016	08:22:51	060	002	001	006	008	001	002
020405	08:23:53.066	08:23:51	060	005	000	000	001	000	000
020405	08:24:53.016	08:24:51	060	001	002	004	003	000	001
020405	08:25:52.996	08:25:51	060	006	006	002	004	003	003
020405	08:26:52.905	08:26:51	060	002	000	000	000	000	005
020405	08:27:52.905	08:27:51	060	001	003	003	001	001	001
020405	08:28:53.226	08:28:51	060	002	000	003	003	001	002
020405	08:29:53.136	08:29:51	060	002	002	002	002	001	003
020405	08:30:53.086	08:30:51	060	006	004	008	007	001	001
020405	08:31:53.026	08:31:51	060	003	001	004	006	001	001
020405	08:32:53.106	08:32:51	060	001	000	001	002	001	001
020405	08:33:53.186	08:33:51	060	002	001	001	002	002	001
020405	08:34:53.036	08:34:51	060	002	001	001	001	001	002
020405	08:35:53.006	08:35:51	060	001	002	003	004	001	001
020405	08:36:53.046	08:36:51	060	004	000	002	002	000	002
020405	08:37:53.006	08:37:51	060	000	000	006	005	003	003
020405	08:38:53.176	08:38:51	060	003	000	002	002	001	003
020405	08:39:53.046	08:39:51	060	005	000	002	008	004	001
020405	08:40:53.076	08:40:51	060	001	001	004	007	003	001
020405	08:41:53.176	08:41:51	060	003	002	000	000	002	003
020405	08:42:53.016	08:42:51	060	001	001	003	001	001	000
020405	08:43:53.286	08:43:51	060	004	003	002	005	001	003
020405	08:44:53.146	08:44:51	060	002	000	001	001	000	004
020405	08:45:53.066	08:45:51	060	003	003	002	004	000	002
020405	08:46:52.154	08:46:51	060	004	000	002	012	002	001
020405	08:47:52.024	08:47:51	060	002	004	001	003	003	003
020405	08:48:52.194	08:48:51	060	004	004	002	000	000	000
020405	08:49:52.124	08:49:51	060	003	000	003	008	005	004
020405	08:50:52.164	08:50:51	060	002	002	001	002	001	002
020405	08:51:52.144	08:51:51	060	001	003	003	002	002	002
020405	08:52:53.106	08:52:51	060	006	004	003	005	003	000
020405	08:53:53.106	08:53:51	060	004	000	002	001	002	003
020405	08:54:53.136	08:54:51	060	004	004	003	007	001	002
020405	08:55:53.026	08:55:51	060	004	000	003	004	001	003
020405	08:56:53.216	08:56:51	060	003	002	000	003	002	003
020405	08:57:53.136	08:57:51	060	005	004	003	002	002	003
020405	08:58:53.146	08:58:51	060	002	002	007	010	001	002
020405	08:59:53.116	08:59:51	060	001	001	001	003	002	002
020405	09:00:53.066	09:00:51	060	005	001	006	003	000	000
020405	09:01:53.046	09:01:51	060	004	000	001	006	004	004
020405	09:02:53.116	09:02:51	060	001	001	000	000	001	001
020405	09:03:53.046	09:03:51	060	001	002	006	007	001	000
020405	09:04:53.226	09:04:51	060	000	001	002	004	002	000
020405	09:05:53.296	09:05:51	060	002	001	001	001	003	001
020405	09:06:53.136	09:06:51	060	003	001	000	004	001	000
020405	09:07:53.136	09:07:51	060	001	002	002	002	002	002
020405	09:08:53.196	09:08:51	060	002	000	000	002	002	001
020405	09:09:53.136	09:09:51	060	003	003	003	000	000	001
020405	09:10:53.236	09:10:51	060	001	002	001	005	001	001
020405	09:11:53.246	09:11:51	060	001	001	002	002	001	000

APPENDIX C.4 – EXCERPT OF THE EXPERIMENT SETUP FILE

```
81          //overall number of runs

Run1LLL     //run identifier

60          // north mean vehicle inter arrival time
60
60
60          // south mean vehicle inter arrival time
60
60
60          // west mean vehicle inter arrival time
60
60

14          // number of lights phases

25    green   green   red     red     black
3     yellow  green   red     red     black
1     red     green   red     red     black
1     red     green   orange  red     black
3     red     yellow  green   red     black
2     red     red     green   red     black
1     red     red     green   red     green
1     red     red     green   orange  green
20    red     red     green   green   black
3     red     red     yellow  yellow  black
2     red     red     red     red     black
1     red     orange  red     red     black
1     orange  green   red     red     black
6     green   green   red     red     black

3600        // seconds of simulation time


Run1MLL     //run identifier

30          // north mean vehicle inter arrival time
30
30
60          // south mean vehicle inter arrival time
60
60
60          // west mean vehicle inter arrival time
60
60

14          // number of lights phases

25    green   green   red     red     black
3     yellow  green   red     red     black
1     red     green   red     red     black
1     red     green   orange  red     black
3     red     yellow  green   red     black
2     red     red     green   red     black
1     red     red     green   red     green
1     red     red     green   orange  green
20    red     red     green   green   black
3     red     red     yellow  yellow  black
2     red     red     red     red     black
1     red     orange  red     red     black
1     orange  green   red     red     black
6     green   green   red     red     black

3600        // seconds of simulation time
```