

Zur Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation

Martin Krückhans

Vollständiger Abdruck der von der Fakultät für Informatik der Universität der Bundeswehr
München zur Erlangung des akademischen Grades

Doktor Ingenieur (Dr.-Ing.)

genehmigten Dissertation.

Gutachter:

1. Univ. Prof. Dr.-Ing. Wolfgang Reinhardt
2. Prof. Dr.-Ing. Reinhard Finsterwalder

Die Dissertation wurde am 10.07.2018 bei der Universität der Bundeswehr München eingereicht und durch die
Fakultät für Informatik am 09.10.2018 angenommen. Die mündliche Prüfung fand am 25.10.2018 statt.

Kurzfassung

In virtuellen Simulationen trainieren reale Personen ihre Fertigkeiten und den Umgang mit nachgebildeten Systemen. Zur Durchführung einer Simulation werden die Eigenschaften und die Umgebung von realen Vehikeln in sogenannten Simulationssystemen rekonstruiert. Zweck einer Simulation ist die ungefährliche und kostengünstige Nachstellung realer Situationen. Da das Trainieren in echten Fahr- oder Flugzeugen stets mit Aufwand, Risiken und Kosten verbunden ist, bieten Simulationssysteme die Möglichkeit, gefahrlos in real wirkenden Umgebungen trainieren zu können.

Ein Bestandteil von Simulationssystemen ist zumeist eine per Computersystem erzeugte synthetische Darstellung der Umgebung. Der Übende bewegt das Fahr- oder Flugzeug virtuell durch eine synthetische 3D-Umwelt. Diese Darstellung der virtuellen Umgebung muss so realistisch wie möglich sein, um die Übertragbarkeit der Übung in die Realität, den sogenannten Trainingseffekt, zu maximieren.

Gerade im militärischen Umfeld ist nicht nur die Beherrschung des Vehikels, sondern auch die Einwirkung auf die synthetische Umwelt ein wichtiger Bestandteil von Übungsszenarien in Simulationen. Die Veränderung von Gebäudedarstellungen, den sogenannten 3D-Gebäudemodellen, durch Waffenwirkungen ist ein notwendiger Prozess in militärischen Simulationen. Die Fähigkeit, Schäden von Gebäuden durch Waffenwirkungen während der Simulation berechnen und darstellen zu können, ist gefordert.

In bestehenden Vorgehensweisen zur Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation wird ein 3D-Gebäudemodell nach einem Schadensereignis durch ein vorprozessiertes und typisiert beschädigtes 3D-Gebäudemodell ausgetauscht. Das unbeschädigte 3D-Gebäudemodell wird durch ein beschädigtes 3D-Gebäudemodell ersetzt, welches einen typischen Schaden enthält. In diesen vorprozessierten 3D-Gebäudemodellen bleiben Parameter, wie der Ort des Einschlags oder die Art der Munition, die erst zur Laufzeit der Simulation feststehen, unberücksichtigt. Der Schaden des typisiert beschädigten 3D-Gebäudemodells repräsentiert nur näherungsweise die Auswirkungen des Schadensereignisses.

In dieser Arbeit wird ein neuer Ansatz zur Handhabung von Gebäudeschäden vorgestellt. Es wird gezeigt, dass die Erzeugung von beschädigten Gebäuden auch zur Laufzeit der Simulation durchgeführt werden kann. Das unbeschädigte 3D-Gebäudemodell wird durch ein beschädigtes 3D-Gebäudemodell ersetzt, welches die konkreten Umstände des Schadensereignisses repräsentiert. Das beschädigte Gebäude entspricht dem Schadensereignis.

Ein Mehrwert dieses Ansatzes ist die Steigerung der Authentizität einer verteilten, virtuellen Simulation. Wenn das dargestellte beschädigte Gebäude dem Schadensereignis entspricht, wird die Simulation vom Übenden als realistischer wahrgenommen. Die Übertragbarkeit der Simulation in die Realität wird verbessert.

Abstract

Virtual simulations are being used by real persons to train and practice their driving, flying or general vehicle controlling skills in replicated simulation systems. In order to achieve this, the characteristics and properties of real vehicles as well as the environment they are operating in are reconstructed. The overall purpose of this reconstructed environment is to reproduce real situations in an inexpensive and safe manner. As training in real vehicles is always associated with effort, risks and costs, simulation systems offer the possibility to train and practice safely in real-world environments.

This synthetic representation of the environment is usually generated by a computer system. The simulation users move the vehicle or aircraft virtually through a synthetic 3d environment. The visualization of this virtual environment must be as realistic as possible in order to maximize the training effect.

Especially in a military context it is not sufficient to visualize this virtual environment in a static way. Besides the pure handling of the vehicle, the influence on and the change of the synthetic environment need to be part of the training scenarios within the simulation. Although ethically questionable, it is from a military viewpoint necessary to understand the impact of weapons and how they determine a change to the building visualization in the simulation. As the handling of building damage in virtual simulation systems is indispensable, the ability to calculate and visualize the damage of buildings during simulations is a key requirement.

This thesis presents a new approach for handling building damage in virtual, distributed simulations. It is shown that realistic damage visualizations, for example through weapon effects or explosions, can both be calculated at runtime and at the same time visualized realistically. In contrast to the current state of the art, the optical damages are not static. Consequently, a virtually damaged building is no longer a visualisation of a preprocessed or a priori known state, but is based on runtime parameters. This will in general allow a more realistic handling of building damages during simulation.

The new approach is implemented in a service-oriented manner as a 3d impact service for distributed simulations. Through the consistent use of spatial data standards, interoperability and reusability are ensured. Finally, the general applicability is verified by using a commercial and professional simulation software product (X-Plane).

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Ziele	3
1.3	Zur Gliederung	4
2	Grundlagen und Stand der Technik	6
2.1	Simulation	6
2.1.1	LVC-Simulationen	6
2.1.2	Virtuelle Simulation	7
2.2	Simulationssysteme für virtuelle Simulationen	7
2.2.1	Simulator	7
2.2.2	Simulationsrechner	8
2.2.3	Sichtsystem	9
2.2.4	Simulationsoperator	9
2.2.5	Simulationssoftware	10
2.3	Anforderungen der militärischen, virtuellen Simulation	10
2.3.1	Verteilte Simulation	11
2.3.2	Simulation der Waffenwirkung	11
2.3.3	3D-Gebäudemodelle	12
2.4	Techniken und Verfahren der verteilten Simulation	13
2.4.1	Simulationsexperiment	13
2.4.2	Joint Exercises	13
2.4.3	Interoperabilität in der verteilten Simulation	15
2.4.4	Bewertung von verteilten Simulationen	18
2.4.5	Fair-Fight Prinzip	18
2.4.6	Modelling and Simulation as a Service	19
2.4.7	High Level Architecture	20
2.4.8	Distributed Interactive Simulation	21
2.4.9	Real-time Platformlevel Reference FOM	21
2.4.10	Weitere Architekturen	22
2.5	Techniken und Verfahren zur Simulation von Waffenwirkungen	22
2.5.1	Klassifikation von Waffeneinsätzen gegen Ziele	22
2.5.2	Waffenwirkungen	23

2.5.3	Simulation von Waffenwirkungen	24
2.5.4	Waffenwirkungen in der virtuellen Simulation	25
2.6	Techniken und Verfahren von 3D-Gebäudemodellen	26
2.6.1	Definition und Eigenschaften	26
2.6.2	Vergleich von Normen und Standards für 3D-Gebäudemodelle	28
2.6.3	3D-Gebäudemodelle auf Basis der Norm ISO 19107	28
2.6.4	3D-Gebäudemodelle auf Basis der Norm ISO/IEC 18023	31
2.6.5	3D-Gebäudemodelle auf Basis des OGC Common Database	31
2.6.6	Visualisierung von 3D-Gebäudemodellen	33
2.7	Stand der Technik - Handhabung von Gebäudeschäden	35
2.7.1	Kurzbeschreibung VIntEL	35
2.7.2	Gebäudeschäden vor der Simulation	36
2.7.3	Gebäudeschäden während der Simulation	38
2.7.4	Ergebnisse des Projekts VIntEL	40
3	Anforderungen und alternative Lösungswege	41
3.1	Definition von beschädigten Gebäuden	41
3.2	Anforderungen zur Handhabung von Gebäudeschäden	42
3.2.1	Authentizität	42
3.2.2	Konsistenz	42
3.2.3	Reaktionszeit	43
3.2.4	Transparenz	44
3.2.5	Integrierbarkeit	45
3.2.6	Bestehende Standards	45
3.3	Bestehende Ansätze von Gebäudeschäden	45
3.3.1	3D-Geoinformationssysteme	45
3.3.2	Realphysikalische Simulationen	47
3.3.3	Videospiele	49
3.3.4	Zusammenfassung	51
4	Ein neues Verfahrens zur Handhabung von Gebäudeschäden	53
4.1	Prozessorientierte Darstellung	53
4.2	Diskussionen und Herleitungen	56
4.2.1	Parameter des Schadensereignisses	56
4.2.2	Berechnung der Schadenstransformation	57
4.2.3	Parameter der Schadenstransformation	58
4.2.3.1	Bestehende Ansätze	58
4.2.3.2	Anforderungen	59
4.2.3.3	Neuer Ansatz	63
4.2.3.4	Erläuterungen zum Ansatz	64
4.2.4	Anwendung der Schadenstransformation	66

4.3	Zusammenfassung - Ein neues Verfahren	69
4.3.1	Voxelisierung	69
4.3.2	Anwendung der Schadenstransformation	71
4.3.3	Retriangulierung	72
5	Implementierungen zum neuen Verfahren	74
5.1	Umsetzung des Verfahrens als 3D-Impactservice	74
5.1.1	Implementierungen zur Voxelisierung	74
5.1.2	Implementierungen zur Berechnung der Schadenstransformation	76
5.1.3	Parameter der Schadenstransformation mittels Standards	78
5.1.4	Implementierungen zur Anwendung der Schadenstransformation	81
5.1.5	Implementierungen zur Retriangulierung	82
5.1.6	Aggregation der Routinen zu einem 3D-Impactservice	83
5.2	Verteilte Evaluierungsumgebung	84
5.2.1	Beschreibung der Evaluierungsumgebung	84
5.2.2	Testreihen und Untersuchung des Laufzeitverhaltens	85
5.2.3	Bewertung der Ergebnisse	87
5.3	Anbindung in eine reale Simulationssoftware	90
5.3.1	Simulationsszenario in X-Plane	90
5.3.2	Implementierung in X-Plane	91
5.3.3	Bewertung der Ergebnisse	92
5.4	Bewertung des neuen Ansatzes hinsichtlich der definierten Ziele	93
5.4.1	Authentizität	93
5.4.2	Konsistenz	95
5.4.3	Transparenz	96
5.4.4	Reaktionszeit	96
5.4.5	Integrierbarkeit	96
5.4.6	Bestehende Standards	96
6	Schlussbemerkungen	98
6.1	Grenzen des Ansatzes und Ausblicke	98
6.1.1	Authentizität	98
6.1.2	Reduktion der Anzahl der Dreiecke des Ausgangszustandes	99
6.1.3	3D-Impactservice auf Basis von Geodatenstandards	99
6.1.4	Erweiterte Anwendbarkeit auf Basis der VIntEL-Architektur	99
6.1.5	Berücksichtigung von Texturen	101
6.2	Fazit	101
	Literaturverzeichnis	102
	Abbildungsverzeichnis	113

Tabellenverzeichnis 114

Abkürzungen und Akronyme

3D-GIS	Dreidimensionale Geoinformationssysteme
AdV	Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland
AI	Artifizielle Intelligenz
API	Application Programming Interface
BAAINBw	Bundesamt für Ausrüstung, Informationstechnik und Nutzung der Bundeswehr
BRep	Boundary Representation
Bw	Bundeswehr
CDB	Common Database
CEP	Circular Error Propable
DGPF	Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling and Simulation Office
DOD	Department of Defense (US-Verteidigungsministerium)
DRM	Data Representation Model
DSEEP	Distributed Simulation Engineering and Execution Process
EDCS	Environmental Data Coding Specification
FEM	Finite-Elemente-Methode
FOM	Federation Object Model
FPS	Frames Per Second

GIS	Geoinformationssystem
GML	Geography Markup Language
HLA	High Level Architecture
IEC	International Electrotechnical Commission
IED	Improvised Explosive Device
ISO	International Organization for Standardization
KdB	Konzeption der Bundeswehr
LoD	Level of Detail
LVC	Live, Virtual and Constructive Simulations
MS	Modellbildung und Simulation
NATO	North Atlantic Treaty Organization
NetOpFü	Vernetzte Operationsführung
OGC	Open Geospatial Consortium
PDU	Protocol Data Units
RPR-FOM	Real-time Platformlevel Reference Federation Object Model
RTI	Run-Time-Infrastructure
SD VIntEL	Systemdemonstrator Verteilte, Integrierte Erprobungslandschaft
SEDRIS	Synthetic Environment Data Representation and Interchange Specification
SES	Synthetic Environment Service
SIG3D	Special Interest Group 3D
Simsys	Simulationssystem
SISO	Simulation Interoperability Standards Organization
SOA	Service Oriented Architecture
SuTBw	Simulations- und Testumgebung der Bundeswehr
TCP/IP	Transmission Control Protocol/Internet Protocol
TENA	Test and Training Enabling Architecture
TSK	Teilstreitkraft

UDP	User Datagram Protocol
VBS	Virtual Battlespace
VEVA	Vorgehensmodell für den Einsatz der VIntEL-Architektur
VS	Voxel Space
VV&A	Verification, Validation and Accreditation
XML	Extensible Markup Language

Kapitel 1

Einleitung

Bei militärischen Streitkräften, beispielsweise bei der deutschen Bundeswehr, wird die Ausbildung von Fahrern oder Piloten durch das Üben in Simulationssystemen ergänzt. „Weil typische Bundeswehraufgaben oft Risiken in sich bergen und an sensiblen technischen Geräten stattfinden, haben Ausbildung und Training an Simulationssystemen in den Streitkräften eine lange Tradition“ [8].

In dieser Arbeit werden militärische Simulationssysteme für bemannte Waffensysteme betrachtet. Drei Anforderungen an militärische Simulationssysteme sind in Kombination für das Thema dieser Arbeit wesentlich:

1. **Verteilte Simulationen:** Verteilte Simulationen bestehen aus vernetzten Simulationssystemen. Die Vernetzung von militärischen Simulationssystemen ist notwendig, um das Training im Kollektiv zu ermöglichen. Eine Situation muss in allen vernetzten Simulationssystemen einer kollektiven Übung visualisiert werden. Der Informationsaustausch zwischen den Simulationssystemen wird benötigt.
2. **Simulation der Waffenwirkung:** In militärischen Simulationen muss die Auswirkung von Waffeneinsätzen berechnet werden. Die Schäden eines Waffeneinsatzes müssen als Feedback für den Übenden visualisiert werden. Dazu gehört auch der Waffeneinsatz gegen Gebäude. Der Waffeneinsatz gegen ein Gebäude kann zu einer Beschädigung des Gebäudes führen.
3. **3D-Gebäudemodelle:** In Simulationssystemen werden virtuelle 3D-Darstellungen von Gebäuden benötigt. In der Simulation werden diese Gebäudedarstellungen verwendet, um für den Übenden in der Simulation den Eindruck einer virtuellen 3D-Landschaft zu vermitteln.

1.1 Motivation

Bereits im Jahr 1995 wurde im *DoD Masterplan für Modellbildung und Simulation (DoD Masterplan M&S)* [53] dargestellt, dass zwar „beeindruckende Darstellungen des Geländes

erzeugt werden können, diese jedoch weitgehend nicht wiederverwendbar sind und die Herstellung dieser Darstellungen sehr viel Zeit, Geld und personelle Ressourcen beansprucht“ [53]. Im Detail wird die Anforderung formuliert, dynamische Veränderungen der synthetischen Umwelt in der verteilten Simulation austauschen zu können (original: „There is a need to [...] support a broader range of capabilities (e.g., to reflect dynamic changes in the environment [...])“ [53]. Auch die Darstellung von Gebäudeschäden ist eine Veränderung der dynamischen Umwelt, die mit der zuvor genannten Anforderung einhergeht.

Obwohl die Publikation des DoD Masterplan M&S mehr als zwei Dekaden zurückliegt, ist bisher nur eine Verfahrenslösung zum Umgang von Gebäudeschäden in virtuellen Simulationen bekannt. Diese Verfahrenslösung ist die Einblendung von vorgerechneten und typisiert beschädigten 3D-Gebäudemodellen. Um den Waffeneinsatz an einem Gebäude simulieren zu können, wird ein Gebäude durch mehreren 3D-Gebäudemodellen unterschiedlicher Beschädigungsgrade, den sogenannten Zustandsstufen (*Level of States (LoS)*), repräsentiert. Jede Zustandsstufe des Gebäudes ist ein 3D-Gebäudemodell eines typisierten Beschädigungsgrades. In Abbildung 1.1 sind exemplarisch drei Zustandsstufen eines Gebäudes dargestellt. Im Fall eines simulierten Waffeneinsatzes, mit einhergehender Beschädigung des Gebäudes, wird für den Übenden das unbeschädigte 3D-Gebäudemodell ausgeblendet und das beschädigte 3D-Gebäudemodell eingeblendet. Für den Übenden entsteht der Eindruck, dass sein Waffeneinsatz den Gebäudeschaden erzeugt hat.

Die Vorhaltung von typisiert beschädigten 3D-Gebäudemodellen ist ein etabliertes Konzept. So existieren mehrere Beispiele für Standards von 3D-Gebäudemodellen in denen die Vorhaltung von Zustandsstufen implementiert ist (vgl. [13, 69, 61]). Auch seitens der Bundeswehr wird das LoS-Konzept, beispielsweise in Forschungsprojekten, wie dem *Systemdemonstrator Verteilte Integrierte Erprobungslandschaft (SD VIntEL)*, für die Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation verwendet [26, 63].

Ein wesentlicher Nachteil vorgerechneter Zustandsstufen ist, dass Wechselwirkungen, die erst während der Simulation auftreten, nur bedingt berücksichtigt werden können. Das Feedback des Waffeneinsatzes für den Übenden kann nur so vielfältig sein, wie es die vorgerechneten Zustandsstufen zulassen. Parameter des Schadensereignisses, wie die Beschussstärke, die Beschussrichtung oder der Ort des Einschlags, die einen realen Gebäudeschaden unmittelbar beeinflussen würden, stehen erst zur Laufzeit der Simulation fest und können nicht in vorgerechneten Zustandsstufen berücksichtigt werden.

An diesem wesentlichen Nachteil der bisherigen Verfahrenslösung setzt diese Arbeit an. Es existiert nach aktuellem Kenntnisstand, auch international, keine Vorgehensweise oder allgemein gültige Lösung zur Berechnung und Darstellung von Waffenwirkungen an 3D-Gebäudemodellen für die verteilte, virtuelle Simulation, die über den Austausch von vorgerechneten 3D-Modellen hinausgeht. Es ist nach aktuellem Stand der Technik nicht möglich, einen Gebäudeschaden während einer verteilten, virtuellen Simulation und in Abhängigkeit zu Parametern eines Schadensereignisses zu berechnen und zu visualisieren. Dieser Stand der Technik ist die Motivation für die vorliegende Arbeit, dass die Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation verbessert werden kann.

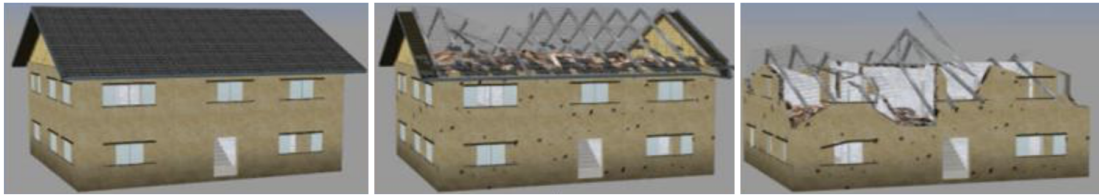


Abbildung 1.1: Darstellung von Gebäudeschäden durch vorgerechnete 3D-Gebäudemodelle, die jeweils einen typisierten Beschädigungsgrad repräsentieren. Abbildungen aus [63].

1.2 Ziele

Das Ziel dieser Arbeit ist die Verbesserung der Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation. Die Handhabung von Gebäudeschäden muss eine Reihe besonderer Merkmale aufweisen um den Anforderungen der verteilten, virtuellen Simulation gerecht zu werden. Ob mit dieser Arbeit eine tatsächliche Verbesserung erreicht wird, ist daran zu bemessen, ob und wie die folgenden Ziele zur Handhabung von Gebäudeschäden umgesetzt werden können. Die Ziele dieser Arbeit sind in Kurzform in Tabelle 1.1 dargestellt und werden im Folgenden erläutert.

Titel des Ziels	Kurzbeschreibung des Ziels
Authentizität	Der Gebäudeschaden muss den spezifischen Eigenschaften der Schadensursache, wie Art und Position der Waffenwirkung, soweit entsprechen, dass die Visualisierung des Gebäudeschadens für den Übenden authentisch ist.
Konsistenz	In den Simulationssystemen einer verteilten Simulation muss ein Gebäudeschaden gleichartig berechnet werden.
Reaktionszeit	Der Gebäudeschaden muss in Echtzeit berechnet und visualisiert werden.
Transparenz	Der Berechnungsvorgang des Gebäudeschadens muss nachvollziehbar und darf nicht proprietär sein.
Integrierbarkeit	Das neue Verfahren muss als Modul in unterschiedliche verteilte Simulationen integrierbar sein.
Bestehende Standards	Etablierte Vorgehensweisen und Standards der verteilten Simulation sind zu berücksichtigen.

Tabelle 1.1: Ziele zur Handhabung von Gebäudeschäden für diese Arbeit.

Authentizität Ein Gebäudeschaden muss nach einer Waffenwirkung in einem Simulationssystem so dargestellt werden, dass sich der Übende weiterhin oder trotzdem mit dem Geschehen in der virtuellen Welt der Simulation identifizieren kann. Der visualisierte Ge-

bäudeschaden muss authentisch sein. Für dieses Ziel muss der Gebäudeschaden den spezifischen Eigenschaften der Schadensursache, wie Beschussstärke, Beschussrichtung oder Ort des Einschlags entsprechen. Es ist ein Verfahren zu entwickeln, das die Schadensursache soweit berücksichtigt, dass authentische Gebäudeschäden berechnet und visualisiert werden können.

Konsistenz Es ist gefordert, dass die Berechnung eines Gebäudeschadens innerhalb einer verteilten Simulation eindeutig ist. Der Berechnungsprozess des Gebäudeschadens muss nach einer Waffenwirkung in allen vernetzten Simulationssystemen gleichartig durchgeführt werden.

Reaktionszeit Die Reaktionszeit muss minimal sein. Die Reaktionszeit ist die Zeit zwischen einer Waffenwirkung und der Visualisierung des Gebäudeschadens. Der Gebäudeschaden muss in Echtzeit berechnet und visualisiert werden.

Transparenz Bei einem dargestellten Gebäudeschaden muss offenkundig sein, welche Berechnungsschritte zu dieser Visualisierung geführt haben. Eine proprietäre Lösung mit unbekanntem Prozessen ist nicht das Ziel dieser Arbeit.

Integrierbarkeit Eine mögliche Verbesserung der Handhabung von Gebäudeschäden darf nicht auf spezielle verteilte Simulationen beschränkt sein. Ein neues Verfahren muss als Modul in unterschiedliche Simulationen mit variierenden Simulationssystemen integriert werden können.

Bestehende Standards Ein weiteres Ziel dieser Arbeit ist die Berücksichtigung bestehender Standards und etablierte Vorgehensweisen. Für verteilte Simulationen existieren diverse Standards und akzeptierte Lösungsstrategien. Diese sind aufzuzeigen und nach Möglichkeit, auch für die Handhabung von Gebäudeschäden, zu adaptieren.

1.3 Zur Gliederung

Im folgenden Kapitel 2 werden die technischen Hintergründe der Simulationen, der 3D-Gebäudemodelle und des Umgangs mit Waffenwirkungen erläutert. Mit dem Kapitel 2 wird ein Grundverständnis der Begriffe und des Themas entwickelt.

Aufbauend auf dem Grundverständnis werden in Kapitel 3 die genannten Ziele dieser Arbeit konkretisiert und zu Anforderungen eines neuen Ansatzes ausformuliert. Diese Anforderungen werden im Abschnitt 3.3 mit alternativen Lösungswegen verglichen.

In Kapitel 4 wird ein neuer Ansatz zur Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation vorgestellt. Die Eigenschaften des neuen Ansatzes werden diskutiert und begründet.

Der vorgestellte Ansatz wird in Kapitel 5 verifiziert. Es wird dargestellt, welche Ziele realisiert werden und ob die Handhabung von Gebäudeschäden tatsächlich verbessert werden konnte.

Im finalen Kapitel 6 werden in einem Ausblick mögliche Themenschwerpunkte von Folgeprojekten und zukünftigen Forschungsaktivitäten skizziert.

Kapitel 2

Grundlagen und Stand der Technik

2.1 Simulation

Eine *Simulation* ist „ein möglichst realitätsnahes Nachbilden von Geschehen der Wirklichkeit. Aus Sicherheits- und Kostengründen ist es für fast alle konkreten Problemkreise notwendig, sie aus der Realität zu lösen und abstrakt zu behandeln“ [23]. Der Begriff der Simulation ist sehr allgemein. Es existieren vielfältige Anwendungsbereiche, in denen ein Geschehen der Wirklichkeit nachgebildet wird.

Eine Spezialisierung des Simulationsbegriffs ist die Bezeichnung der sogenannten *human-in-the-loop-Simulationen* (wörtlich übersetzt: „Mensch-in-der-Schleife-Simulation“). „Eine Simulation wird als human-in-the-loop bezeichnet, wenn ein oder mehrere menschliche Bediener in direkter Steuerung der Simulation eingesetzt werden“ (frei übersetzt aus [27]). Ein Zweck von human-in-the-loop-Simulationen ist das Trainieren der menschlichen Bediener.

2.1.1 LVC-Simulationen

Im militärischen Umfeld wird, neben den Spezialisierungen der human-in-the-loop-Simulationen, die Definition der sogenannten LVC-Simulationen verwendet. Das Akronym *LVC* beschreibt nach dem Stichwortverzeichnis des amerikanischen Verteidigungsministeriums für Modellbildung und Simulation [27] die drei Simulationsarten *live*, *virtuell* und *konstruktiv*.

- *Live Simulationen*: Reale Personen \longleftrightarrow Reale Systeme
- *Virtuelle Simulationen*: Reale Personen \longleftrightarrow Virtuelle Systeme
- *Konstruktive Simulationen*: Virtuelle Personen \longleftrightarrow Virtuelle Systeme

L- und V-Simulationen sind auf Grund der Abhängigkeit von menschlichen Interaktionen (reale Personen) der Gruppe der human-in-the-loop-Simulationen zuzuordnen [88]. Hierbei agieren stets reale Personen und können den Simulationsverlauf aktiv beeinflussen. Art und Weise sowie Kadenz und Dauer der Beeinflussung sind dabei nicht weiter festgelegt,

wobei häufig eine Interaktion zu jedem Zeitschritt der Simulation möglich ist. L- und V-Simulationen unterscheiden sich darin, ob die Simulation im originalen System oder in einer Nachbildung stattfindet. Wenn eine reale Person ein originales System kontrolliert, handelt es sich um eine Live Simulation¹. Ist das System hingegen nachgebildet, spricht man von einer virtuellen Simulation [95]. Diese Arbeit fokussiert virtuelle Simulationen.

2.1.2 Virtuelle Simulation

„In einer *virtuellen Simulation* agieren reale Personen in nachgebildeten, also virtuellen, Systemen“ (frei übersetzt aus [95]). „Dazu werden reale Systeme wie Fahrzeuge in einer synthetischen Umgebung (virtuelle Umwelt) nachgestellt, damit reale Personen ihre Fertigkeiten im Umgang mit dem System trainieren können“ (frei übersetzt aus [27]). Virtuelle Simulationen werden auch im zivilen Bereich, beispielsweise bei der Ausbildung von Polizeikräften, eingesetzt.

Die originalen Systeme, also die nachzubildenden Fahrzeuge, Flugzeuge oder Helikopter, werden als *Realsysteme* bezeichnet. Die nachgebildeten Systeme in denen die Simulation, also das Trainieren, durchgeführt wird, sind die sogenannten *Simulationssysteme*. Eine Übersicht dieser Begriffe ist am Beispiel von zivilen Simulationssystemen in Abbildung 2.1 dargestellt. Die Begriffe gelten analog für militärische Real- und Simulationssysteme.





Realität		Nachgebildete Realität	
reales System (<i>Realsystem</i>)	reale Situation	simuliertes System (<i>Simulationssystem</i>)	simulierte Situation (<i>Simulation</i>)
			

Tabelle 2.1: Übersicht der Simulationsbegriffe am Beispiel von zivilen Simulationssystemen. Abbildungen aus [4, 7, 75] (verändert).

2.2 Simulationssysteme für virtuelle Simulationen

Ein Simulationssystem besteht aus unterschiedlichen Komponenten. Die typischen Bestandteile eines Simulationssystems für virtuelle Simulationen sind in Abbildung 2.1 dargestellt. Die Bestandteile werden im Folgenden erläutert.

2.2.1 Simulator

Der *Simulator* ist ein Bestandteil eines Simulationssystems. In einem Simulator sind elementare Komponenten eines Realsystems, beispielsweise die Cockpitnachbildung eines Flug-

¹Der einzige Unterschied von Live Simulationen zu realen Situationen im militärischen Kontext ist, dass in der Simulation keine echten Gegner vorhanden sind [95].

zeugs oder die Fahrerkabine eines Fahrzeugs, nachgebildet. Steuerelemente, Kontrollinstrumente und Monitore von Simulatoren sind häufig in voll beweglichen Projektionsräumen installiert (vgl. Abbildung 2.1).

Ein Simulator ist kein Computersystem. Zwar sind die Monitore oder Bedienungsinstrumente ein Bestandteil des Simulators, die Steuerung dieser Instrumente und Bildschirme werden jedoch von Simulationsrechner und Sichtsystem vorgenommen, welche nachfolgend erläutert werden.



Abbildung 2.1: Bestandteile eines Simulationssystems. Abbildung aus [7] (Ausschnitt).

2.2.2 Simulationsrechner

Der *Simulationsrechner* ist ein Computersystem, das die funktionale Nachbildung des Realsystems steuert. Der Simulationsrechner ist ein Bestandteil eines Simulationssystems. Zu Trainings- und Ausbildungszwecken oder zur Entscheidungsunterstützung komplexer Szenarien ist eine realistische Reaktion des Simulationssystems auf Aktionen des Übenden notwendig. Das Simulationssystem reagiert auf die Steuereingaben des Übenden im Simulator und berechnet Rückkopplungen für den Übenden. In einem Fahrsimulator muss sich beispielsweise die angezeigte Geschwindigkeit im nachgebildeten Tachometer erhöhen, je länger das Gaspedal betätigt wird. Die simulierte Geschwindigkeit muss berechnet werden. Je nachdem welcher Gang eingelegt wurde, ist dies keine triviale Berechnung, sondern eine komplexe Funktion, die von diversen Einflüssen, wie dem simulierten Gewicht des Fahrzeugs, dem simulierten Gefälle der Straße oder Ähnlichem, abhängig ist. Die Berechnung

von Parametern, wie der simulierten Geschwindigkeit, wird durch den Simulationsrechner durchgeführt.

2.2.3 Sichtsystem

Das *Sichtsystem* ist ein weiterer Bestandteil eines Simulationssystems. Es wird zur Erzeugung von visuellen Rückkopplungen benötigt. Die Aufgabe des Sichtsystems besteht darin, die virtuelle Umwelt zu visualisieren und fortlaufend zu aktualisieren. Die Visualisierung der virtuellen Umgebung (synthetischen Umwelt) in einem Fahrsimulator muss durch die simulierte Geschwindigkeit ständig angepasst werden. Betätigt der Übende das Gaspedal (Aktion des Nutzers), berechnet der Simulationsrechner fortlaufend eine aktuelle Geschwindigkeit. Die aktuelle Geschwindigkeit wird dem Übenden nicht nur über den Tachometer, sondern auch über eine virtuelle Bewegung durch die synthetische Umwelt dargestellt (Reaktion des Systems). Es entsteht für den Übenden der Eindruck, als würde er das Fahrzeug tatsächlich steuern. Die Berechnung der Geschwindigkeit ist Aufgabe des Simulationsrechners. Die visuelle Rückkopplung für den Übenden durch eine angepasste Darstellung der synthetischen Umwelt ist Aufgabe des Sichtsystems. Ein Beispiel für ein Simulationssystem mit Sichtsystem und Simulator ist exemplarisch in Abbildung 2.2 dargestellt.

Die Visualisierung der synthetischen Umwelt in einem Sichtsystem muss zu jedem Zeitschritt der Simulation neu berechnet werden, da zumeist Situationen in Bewegung dargestellt werden (Fahrt, Flug, etc.). Diese Neuberechnung der perspektivischen Darstellung wird als *Bildwiederholung* bezeichnet. Die Anzahl der Bildwiederholungen in einer bestimmten Zeit, beispielsweise einer Sekunde, wird als *Bildwiederholfrequenz* (Englisch: *Frames per Second (FPS)*) bezeichnet (vgl. [95]). Im Idealfall ist die Bildwiederholfrequenz des Sichtsystems höher als die wahrnehmbare Aufnahme­frequenz des menschlichen Auges. Andernfalls nimmt der Trainierende ein unrealistisches „Ruckeln“ der Simulation wahr.

2.2.4 Simulationsoperator

Der Ausbilderplatz für den sogenannten *Simulationsoperator* (vgl. Abbildung 2.1 unten links) ist ein weiterer typischer Bestandteil eines Simulationssystems. Vom Ausbilderplatz kann der Simulationsoperator sowohl das Sichtsystem, also auch den Simulationsrechner steuern. Um spezifische Handlungsabläufe, beispielsweise Gefahrensituationen, zu trainieren, kann der Simulationsoperator bestimmte Übungsszenarien in die Simulation einspielen. Im Fall eines Fahrzeugsimulationssystems wird beispielsweise durch Reduktion der simulierten Geschwindigkeit ein realistischer Getriebeschaden simuliert, um die Kenntnisse und Reaktionen eines Fahrers in solchen Situationen zu schulen. Zur Einspielung dieser Übungsszenarien werden die Berechnungen des Simulationsrechners und die Darstellungen des Sichtsystems bewusst manipuliert.



Abbildung 2.2: Sichtsystem, Simulator und trainierende Personen eines Helikoptersimulationssystems. Abbildung aus [21] (Ausschnitt).

2.2.5 Simulationssoftware

Simulationssysteme die als Softwareprogramme auf nur einem Computersystem ausgeführt werden, werden als sogenannte *Simulationssoftware* oder *Simulationsprogramme* bezeichnet. Neben den komplexen Simulationssystemen, wie sie zuvor dargestellt wurden, existieren vielfältige kompakte Simulationssysteme, die auf nur einem Computersystem ausgeführt werden. Der Simulator einer Simulationssoftware besteht aus den herkömmlichen Ein- und Ausgabegeräten eines Desktopcomputers, wie Tastatur, Maus, Joystick und Monitor. Simulationsrechner und Sichtsystem sind keine getrennten Computersysteme, sondern Bestandteil des Simulationsprogramms. Einen spezifischen Ausbilderplatz gibt es nicht. In Simulationsprogrammen wird eine möglichst realistische Abbildung von realen Phänomenen umgesetzt [97]. Ein Beispiele für eine kommerzielle Simulationssoftware, die für diese Arbeit von Bedeutung ist, ist das Programm X-Plane [97]. X-Plane ist eine Flugsimulationssoftware, die sich durch hohen Realismus und Erweiterbarkeit (*Customizing*) auszeichnet [95].

2.3 Anforderungen der militärischen, virtuellen Simulation

Durch militärische Simulationen wird die Ausbildung von Piloten oder Fahrern bemannter Waffensysteme in den Streitkräften unterstützt. Es gibt „[...] kein Waffensystem der Bundeswehr, das nicht in Form eines [...] Simulationssystems nachgebildet wird“ [22].

Die Simulationssysteme bemannter Waffensysteme weisen eine Vielzahl unterschiedlicher Beschaffenheiten und Befähigungen auf. Für diese Arbeit sind unter der Vielzahl der Spezifika militärischer Simulationssysteme, drei Anforderungen relevant. Dies sind die Notwendigkeiten **(1) verteilte Simulationen** durchzuführen, die **(2) Waffenwirkungen** zu simulieren und **(3) 3D-Gebäudemodelle** darzustellen. Die drei Anforderungen werden in

Anforderungen militärischer Simulationen für diese Arbeit	Kurzbeschreibung	Techniken und Verfahren
Verteilte Simulation	Unterabschnitt 2.3.1	Abschnitt 2.4
Simulation der Waffenwirkung	Unterabschnitt 2.3.2	Abschnitt 2.5
3D-Gebäudemodelle	Unterabschnitt 2.3.3	Abschnitt 2.6

Tabelle 2.2: Inhaltliche Übersicht der nachfolgenden Abschnitte.

den folgenden drei Unterabschnitten kurz erläutert. Eine jeweils ausführliche Darstellung der Techniken und Verfahren zu jeder der drei Eigenschaften erfolgt in den anschließenden Abschnitten.

2.3.1 Verteilte Simulation

Mit vernetzten militärischen Simulationssystemen werden *verteilte Simulationen* durchgeführt. Eine Simulation gilt als verteilt, wenn sie zeitgleich in mehreren vernetzten Simulationssystemen ausgeführt wird. Jede partizipierende Einheit, beispielsweise jedes Simulationssystem, wird als *Föderat* und die Systemarchitektur aller Einheiten als eine *Föderation* bezeichnet. Häufig findet sich für eine verteilte Simulation auch die Bezeichnung des *föderierten Systems*. Techniken und Prinzipien der verteilten Simulation werden im Abschnitt 2.4 näher erläutert. Ein Beispiel für eine verteilte, virtuelle Simulationen sind die sogenannten Joint Exercises die in Abschnitt 2.4.2 dargestellt werden.

Allgemein ist die *Verteilung* von militärischen Systemen nicht auf Simulationen beschränkt. So steht die Bezeichnung des „*Network-Centric Warfare* (deutsch: netzwerkzentrierte Kriegführung) für ein militärisches Konzept, das durch die Vernetzung von Aufklärungs-, Führungs- und Wirksystemen Informationsüberlegenheit herstellen und somit [...] eine teilstreitkräfteübergreifende Überlegenheit in der gesamten Reichweite militärischer Operationen garantieren soll“ [95]. Auch seitens der Bundeswehr wird die sogenannte Fähigkeit zur *Vernetzten Operationsführung (NetOpFü)* propagiert [39]. Das primäre Ziel, welches beispielsweise im Konzept zur *SuTBw (Simulations- und Testumgebung der Bundeswehr)* dargestellt ist, ist „die Bereitstellung der notwendigen Fähigkeiten, die für die Vernetzung von realen und simulierten Systemen notwendig sind“ [39, 40]. Zusammenfassung: **Die Vernetzung von Simulationssystemen zur Durchführung von verteilten Simulationen ist gefordert.**

2.3.2 Simulation der Waffenwirkung

Mit der Simulation von Waffensystemen geht einher, dass nicht nur die Handhabung von Vehikeln, sondern auch der Umgang mit deren Bewaffnungen trainierbar sein muss. Der Bediener einer Simulation erwartet nach dem Abschuss oder der Betätigung einer Waffe eine Reaktion des Simulationssystems. Er benötigt zu Übungszwecken ein Feedback, ob und welchen Schaden er mit der Waffe erzeugt hat. Dieses Feedback muss im Simulati-

onsrechner berechnet und dem Übenenden per Sichtsystem² zur Verfügung gestellt werden. **Es ist notwendig, dass die Wirkung einer Waffe während einer Simulation im Sichtsystem visualisiert wird.**

2.3.3 3D-Gebäudemodelle

In den Sichtsystemen werden virtuelle 3D-Modelle der Gebäude, die sogenannten 3D-Gebäudemodelle, dargestellt. Ein *3D-Gebäudemodell* ist, im Kontext dieser Arbeit, ein dreidimensionales, virtuelles Abbild der äußeren Oberfläche eines Gebäudes³. Die 3D-Gebäudemodelle werden auf dem Terrain platziert, um für den Bediener den Eindruck einer virtuellen 3D-Landschaft zu vermitteln [88] (vgl. Darstellung in Abbildung 2.2).

In einem Simulationssystem wird dem Bediener durch das Sichtsystem eine synthetische Umwelt dargestellt. Die *synthetische Umwelt* (original: Synthetic Environment) „ist eine Menge von Datenelementen, die die Umgebung definieren, in der eine Simulation durchgeführt wird“ [27]. Die wichtigsten Bestandteile einer synthetischen Umwelt nach Tolk et al. [88] sind

- das Terrain in der Form eines texturierten 3D-Geländemodells,
- virtuelle 3D-Modelle zur Darstellung von Gebäuden,
- und Vektordaten zur Repräsentation von zweidimensionalen geographischen Informationen (für diese Arbeit nicht relevant).

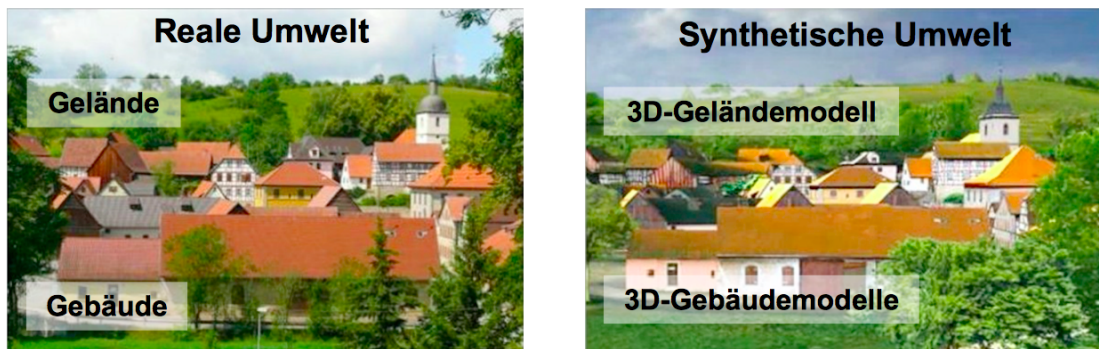


Abbildung 2.3: Vergleich von realer Umwelt und synthetischer Umwelt eines Simulationssystems aus [10] (Beschriftungen ergänzt).

Gelände und Gebäude der Realität werden im Simulationssystem als synthetisches 3D-Geländemodell und synthetisches 3D-Gebäudemodell dargestellt (vgl. Abbildung 2.3). Der

²Andere Formen des Feedbacks, wie Töne oder Vibrationen, werden in dieser Arbeit nicht thematisiert.

³Im Gegensatz zu infanteristischen Simulationen, sind bei der Simulation von bemannten Waffensystemen die Innenräume von Gebäuden von geringem Interesse. Der Einsatz von Fahrzeugen, Flugzeugen oder Helikoptern finden nur in minimalem Umfang innerhalb von Gebäuden statt.

Unterschied zwischen realer und synthetischer Umwelt ist, wie in Abbildung 2.3 deutlich wird, nur schwer erkennbar. Auch durch die 3D-Gebäudemodelle entsteht für den Übenden eine realitätsnahe Darstellung der Wirklichkeit. **3D-Gebäudemodelle werden für eine realistische Nachbildung der Wirklichkeit im Sichtsystem benötigt.**

2.4 Techniken und Verfahren der verteilten Simulation

„What was a good rule for the standalone operation, becomes a bad one in the joint context.“

Andreas Tolk [87]

Verteilte Simulationen gelten auch im militärischen Kontext als komplex [71, 88]. Zur Durchführung von verteilten Simulationen wurden und werden eine Vielzahl von Techniken und Verfahren entwickelt. Ausgewählte und für diese Arbeit relevante Techniken und Verfahren der verteilten Simulation werden in diesem Abschnitt betrachtet.

2.4.1 Simulationsexperiment

Der Ablauf einer verteilten Simulation wird auch als *Simulationsexperiment* bezeichnet. In Abbildung 2.4 sind die verschiedenen Zeiten und Phasen eines Simulationsexperimentes dargestellt. Unter der Voraussetzung, dass eine gemeinsame Datenbasis Verwendung findet (generiert wurde), werden die beteiligten Simulationssysteme während der Initialisierungsphase aus dieser Datenbasis initialisiert, d.h. mit den zu nutzenden Daten versorgt. Im anschließend beginnenden Simulationsverlauf erfolgt die Kommunikation zwischen den Simulationssystemen zu eindeutigen und diskreten Zeitpunkten der Simulationszeit. Nach n Zeitschritten wird das Simulationsexperiment beendet.

2.4.2 Joint Exercises

Eine verteilte Simulation mit Simulationssystemen unterschiedlicher Realsysteme wird auch als *Joint Exercise* [88] (frei übersetzt: „Gemeinsame Übung“) bezeichnet. In realen militärischen Einsatzszenarien agieren Fahrzeuge, Flugzeuge oder Helikopter selten alleine. Einsätze mit verschiedenen Waffensystemen sind die Regel. Dementsprechend hoch ist der Bedarf, Simulationssysteme zum Training im Kollektiv zu vernetzen. Das Beispiel einer Joint Exercise mit heterogenen Simulationssystemen ist in Abbildung 2.5 dargestellt. In der Joint Exercise in Abbildung 2.5 wird eine verteilte Simulation von NH90 (Hubschrauber), Eurofighter Typhoon (Kampfflugzeug) und Puma (Schützenpanzer) skizziert. Für die Simulation müssen die jeweiligen Simulationssysteme der Realsysteme in einer Joint Exercise vernetzt werden.

In einer Joint Exercise können sich die Simulationssysteme stark voneinander unterscheiden. Ein Waffensystem, beispielsweise ein Kampfhubschrauber, ein Kampfflugzeug oder ein Schützenpanzer, hat spezielle Eigenschaften, die sich von anderen Flug- oder Fahrzeugtypen unterscheiden. Die Realsysteme sind heterogen. Aus der Heterogenität der Realsysteme

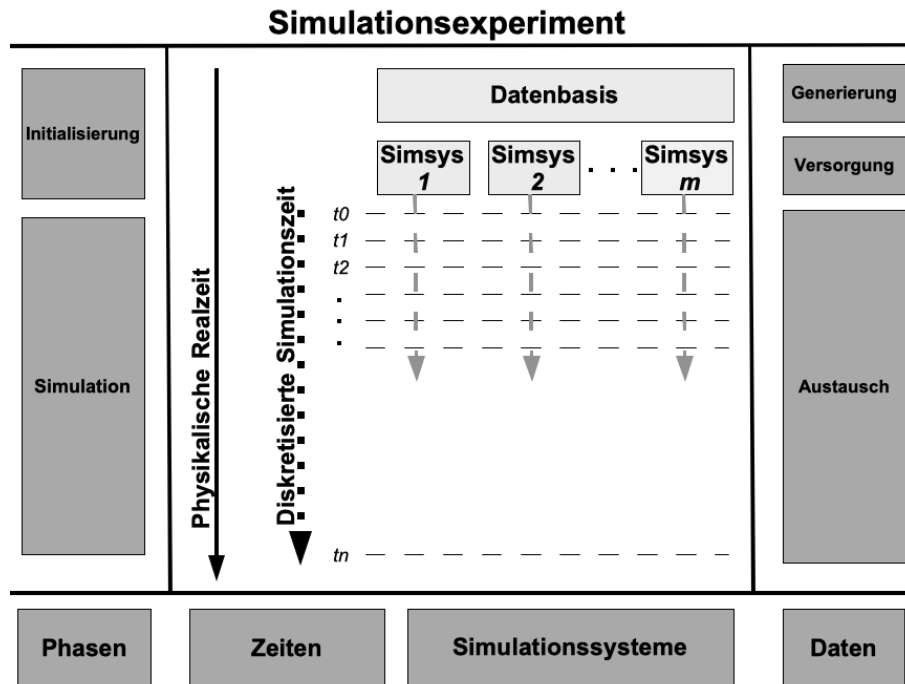


Abbildung 2.4: Ablauf einer verteilten Simulation.

folgt, dass auch die jeweiligen Simulationssysteme spezifische Eigenschaften aufweisen und sich grundlegend voneinander unterscheiden. So sind „die in der Bundeswehr vorhandenen Simulationssysteme oftmals Stand-Alone-Systeme, die für jeweils einen spezifischen Zweck [...] entwickelt wurden. [...] Die geforderte Vernetzung wurde bei der Beschaffung noch nicht berücksichtigt“ [26]. Aus diesem Grund gilt die Vernetzung von Simulationssystemen und die Durchführung von verteilten Simulationsexperimenten als anspruchsvoll und hoch komplex [71].

Vor einem Simulationsexperiment mit heterogenen Simulationssystemen müssen weitreichende Absprachen, Anpassungen und Entwicklungen durchgeführt werden [71]. Für den Prozess der Absprachen vor einer verteilten Simulation *-wer, was, wie während der Simulation simuliert und austauscht-* existieren standardisierte Vorgehensweisen. Bei den Vorgehensweisen handelt es sich um die Beschreibung tatsächlicher Absprachen zwischen Programmierern oder Simulationsingenieuren und nicht um automatisierte Prozesse. Ein Beispiel für eine standardisierte Vorgehensweise zur Vorbereitung und Durchführung von verteilten Simulationen ist der Standard des *Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)* [16]. In DSEEP werden die Prozesse und Prozeduren, denen Benutzer von verteilten Simulationen folgen sollen, um ihre Simulationen zu entwickeln und auszuführen, beschrieben (frei übersetzt aus [16]). Eine Erweiterung des DSEEP für die Bundeswehr ist das *Vorgehensmodell für den Einsatz der VIntEL-Architektur (VEVA)* (vgl. [71]).

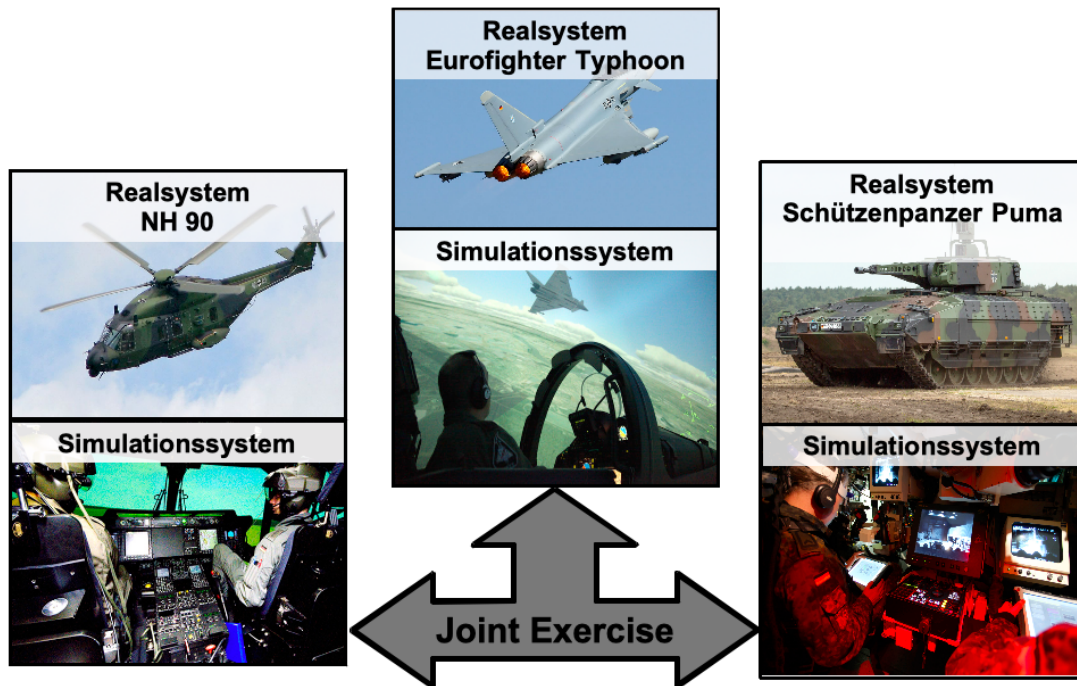


Abbildung 2.5: Beispiel einer verteilten Simulation mit heterogenen Simulationssystemen (Joint Exercise). Jeweils mit Darstellung des Realsystems. Einzelabbildungen aus [8, 41, 50, 95] (Beschriftungen ergänzt).

2.4.3 Interoperabilität in der verteilten Simulation

In der militärischen Simulation wird die *Interoperabilität* nach dem *DoD Masterplan M&S* [53] als „die Fähigkeit einer [...] Simulation definiert, Dienstleistungen zu erbringen und Dienstleistungen von anderen [...] Simulationen zu akzeptieren um die so ausgetauschten Dienste effektiv und gemeinsam zu nutzen (original: „The ability of a [...] simulation to provide services to, and accept services from, other [...] simulations, and to use the services so exchanged to enable them to operate effectively together.“ [53]).

Die Frage, wie Interoperabilität in der verteilten Simulation festgestellt und gemessen werden kann, ist Gegenstand mehrerer Forschungsaktivitäten (vgl. [37, 89, 90]). Ein Ansatz zur Klassifikation von Interoperabilität sind stufenartige Modelle. Die Arbeiten von Turnitsa [90] und Tolk [87, 88, 89] können im Zusammenhang zur stufenartigen Klassifikation von Interoperabilität zwischen Simulationssystemen als maßgebend angesehen werden. Ihr auf sieben Level basierender Ansatz, *Levels of Conceptual Interoperability Model (LCIM)*, zur Klassifikation von Interoperabilität, wird auch in Studien und Leitfäden der Bundeswehr aufgegriffen (vgl. [71, 72]).

Die Interoperabilität einer verteilten Simulation kann nach dem LCIM bewertet werden. Jedes Level beschreibt eine Art des Informationsaustausches zwischen Simulationssystemen. Werden in einer verteilten Simulation alle Eigenschaften des entsprechenden Levels

erfüllt, gilt diese Interoperabilitätsstufe für die Simulation als erreicht. Level 0 steht im LCIM für die geringste und Level 6 für die maximale Interoperabilität. Die sieben Level (vgl. Abbildung 2.6) werden im Folgenden dargestellt.

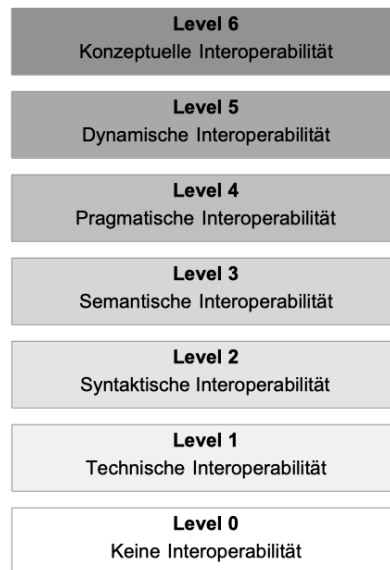


Abbildung 2.6: Die Stufen der Interoperabilität von Simulationssystemen nach Tolck und Turnitsa [87, 88, 89, 90] (frei übersetzt).

Level 0 - Keine Interoperabilität Zwei Simulationssysteme, die keine Schnittstelle zum Austausch von Daten oder Informationen anbieten und somit isoliert sind, gelten als nicht interoperabel. Das Level *Keine Interoperabilität* definiert diesen Zustand und umfasst sowohl die absente als auch die nicht herzustellende Möglichkeit einer Kommunikationsinfrastruktur zwischen den Systemen. Das Kriterium für dieses Level ist bereits verletzt, wenn zwei Simulationssysteme dasselbe Computernetzwerk wie das Internet nutzen. Auch wenn der konkrete Austausch von Informationen durch technische Gegebenheiten praktisch nicht umsetzbar ist, gilt die theoretische Kommunikationsmöglichkeit per Internetanschluss bereits als höheres Level der Interoperabilität.

Level 1 - Technische Interoperabilität Wenn die abstrakte Möglichkeit eines Signalaustausches per elektronischer Datenverarbeitung zwischen Simulationssystemen gegeben ist, sind die Bedingungen für die *technische Interoperabilität* erfüllt. Dies umfasst neben dem Austausch des Signals auch eine gegenseitige Interpretierbarkeit auf technischer Ebene. Die Verwendung von Netzwerkprotokollen wie *IPX*, *TCP/IP* oder *UDP*⁴ zum Austausch von abstrakten Datenpaketen, aber auch der Austausch durch sogenannte etablierte Netzwerkmodelle wie *FTP* oder das *HTTP*⁵ sind Bestandteil der technischen Interoperabilität.

⁴Internetwork Packet Exchange (IPX), Transmission Control Protocol/Internet Protocol (TCP/IP), User Datagram Protocol (UDP)

⁵File Transfer Protocol (FTP) und Hyper Text Transfer Protocol (HTTP)

Level 2 - Syntaktische Interoperabilität Besteht über die Netzwerkprotokolle hinaus Einigkeit über verwendete Metastrukturen, beispielsweise, ob die Informationen als Klartext oder binär übertragen werden, liegt mindestens das Level der *syntaktischen Interoperabilität* vor.

Level 3 - Semantische Interoperabilität Wenn *semantische Interoperabilität* besteht, ist ein gleiches Verständnis des Informationsaustausches (original: „common understanding of information exchange“ [87]) zwischen den Simulationssystemen hergestellt. Auf diesem Level können Informationen durch vordefinierte Begriffe (oder Codewörter), die für alle Simulationssysteme bekannt sind, strukturiert werden. Ein Beispiel: In einer verteilten Simulation wird durch den Simulationsoperator eine simulierte Wetteränderung eingespielt, beispielsweise ein aufziehendes Gewitter. Beim Austausch der Wetteränderung könnten die Bestandteile des Gewitters, wie Windrichtung oder Position der Gewitterzelle, über Codewörter wie „*\$GW_RI*“ (Windrichtung), „*\$GW_POS*“ (Aktuelle Position des Gewitters) gekennzeichnet sein. Anhand dieser Codewörter oder Präfixes werden die ausgetauschten Informationen von den Simulationssystemen verstanden. Die identische Interpretation der Nachrichten oder allgemeine Eindeutigkeit der Information über die Föderation hinaus ist dadurch jedoch nicht gewährleistet.

Level 4 - Pragmatische Interoperabilität Die Bedingungen für *pragmatische Interoperabilität* sind erfüllt, wenn die Eindeutigkeit von Methoden und Funktionen der Simulationssysteme gewahrt ist. Teil der vorherigen semantischen Interoperabilität ist der reine Austausch von Informationen. Die Interpretation und Auswertung der ausgetauschten Informationen obliegt jedoch dem einzelnen Simulationssystem. Mit der pragmatischen Interoperabilität wird zusätzlich die Eindeutigkeit der Informationsinterpretation hergestellt. Im Beispiel des aufziehenden Gewitters ist der Informationsaustausch von Windrichtung und Position des Gewitters ein Teil der semantischen Interoperabilität. Wie diese ausgetauschten Informationen im Sichtsystem für den Übenenden dargestellt werden, ist noch nicht eindeutig. Erst auf dem Level der pragmatischen Interoperabilität würde durch die Eindeutigkeit von Methoden und Funktionen eine visuell identische Darstellung des Gewitters ermöglicht.

Level 5 - Dynamische Interoperabilität Die *dynamische Interoperabilität* ist eine Erweiterung des pragmatischen Levels um den Aspekt der zeitlichen Eindeutigkeit. Die Bedingungen der dynamischen Interoperabilität sind verletzt, wenn im Beispiel des Gewitters die Reihenfolge von Donner- oder Blitzereignissen nicht synchron sind.

Level 6 - Konzeptuelle Interoperabilität In einer verteilten Simulation ist die Bedingung der *konzeptuellen Interoperabilität* erfüllt, wenn Föderaten die Eigenschaften, Prozesse und Wechselwirkungen anderer Simulationssysteme vollständig berücksichtigen (original: „*Interoperating systems at this level are completely aware of each others information,*

processes, contexts, and modeling assumptions“) [89].

Dies geht im Unterschied zur pragmatischen Interoperabilität über die Eindeutigkeit von Methoden und Funktionen hinaus und beinhaltet auch die gegenseitige Kenntnis und den gegenseitigen Zugriff auf Kennzahlen, Bedingungen, logische Abläufe und Fehlertoleranzen. In der Praxis ist konzeptuelle Interoperabilität zwischen unterschiedlichen Simulationssystemen nur schwer realisierbar, da kaum ein System den uneingeschränkten Zugriff auf sämtliche Eigenschaften zulässt.

2.4.4 Bewertung von verteilten Simulationen

Wann ist eine Simulation richtig oder falsch? Diese Frage der Bewertung von Simulationen wird unter dem Akronym *VV&A* (*Verifikation, Validierung and Akkreditierung*) zusammengefasst. VV&A steht dabei nicht für einen Standard oder eine etablierte Methode, sondern definiert den begrifflichen Rahmen der allgemeinen Anwendbarkeit einer Simulation. Die logische Überprüfung von Simulationsverläufen fällt dabei genauso unter VV&A, wie die Übertragbarkeit der trainierten Situationen in die Realität.

Den einzelnen Definitionen zur Verifikation, Validierung und Akkreditierung, wie sie beispielsweise in [27] zu finden sind, verbindet, dass eine eindeutige Bewertung von Simulationen nicht möglich ist. Es kommt immer auf den Zusammenhang an, was zu welchem Zweck und für wen in der Simulation nachgebildet werden sollte. Die Bewertung, ob eine Simulation richtig oder falsch ist, ist vom Auge des Betrachters abhängig. Mit anderen Worten: Es gibt weder falsche noch richtige Simulationen. Sie können nur unter bestimmten Voraussetzungen für eine bestimmte Nutzergruppe als richtiger oder weniger richtig wahrgenommen werden.

2.4.5 Fair-Fight Prinzip

Das *Fair-Fight Prinzip* ist sowohl ein Anforderungsprofil als auch ein Qualitätsmerkmal in der verteilten Simulation. In Videospielen ist die Beschreibung eines *Cheat* („Betrug oder Schwindel“ [95]) eine äquivalente Bezeichnung für die Verletzung des Fair-Fight Prinzips der Simulation.

„*Fair-Fight* zwischen zwei Simulationssystemen besteht, wenn die Unterschiede in der Abbildung der Realität in den Simulationsmodellen nicht zu einer modellimmanenten Übervorteilung eines der beiden Simulationssysteme und damit zu unrealistischen Simulationsergebnissen führt“ [27, 72, 91]. Das Fair-Fight Prinzip ist so lange gewahrt, bis ein Simulationssystem einen systembedingten Vor- oder Nachteil gegenüber einem zweiten System erfährt.

In der Praxis ist Fair-Fight nicht immer zu realisieren. Zwar gilt das Gebot, das Fair-Fight Prinzip zu wahren, dennoch ist ein exaktes Fair-Fight bei vielen Simulationsexperimenten technisch unmöglich. „Es gibt kein absolutes Fair-Fight“ [52]. So wird Fair-Fight auch als Pseudonym für eine exakte, fehlerfreie Simulation verwendet, die es so gut wie möglich zu realisieren gilt. Gerade vor dem Hintergrund, dass in verteilten Simulationen

teilweise Simulationssysteme unterschiedlicher Herstellungsjahrzehnte interagieren müssen, ist die Einhaltung des Prinzips schwierig. Allein die Auflösungen von Bildschirmen oder Projektoren sind bei einem gewissen Altersunterschied zweier Systeme durch die ständig fortschreitende technische Entwicklung nicht identisch. Theoretisch würde bereits eine minimal höhere Auflösung des Sichtsystems von A gegenüber dem Sichtsystem von B zu einem Vorteil von A führen, so dass das Fair-Fight Prinzip verletzt wäre. Die Notwendigkeit einer vernetzten Simulation mit A und B kann jedoch in der Praxis höher bewertet werden als die Einhaltung des Fair-Fight Prinzips. Fair-Fight ist daher ein theoretisches Anforderungsprofil, dessen Einhaltung in der Praxis für den Einzelfall separat definiert werden muss. [72]

2.4.6 Modelling and Simulation as a Service

Das Konzept des *Modelling and Simulation as a Service (MSaaS)* beschreibt das Vorgehen, potenziell redundante Funktionen einer verteilten Simulation zu zentralisieren. Da bestimmte Rechenvorgänge in mehreren Simulationssystemen durchgeführt werden müssten, ist es sinnvoll, diese Funktionen als Service zentral zur Verfügung zu stellen. Gustavson et al. [31] stellten bereits im Jahr 2004 mit *Moving towards a Service-Oriented Architecture (SOA) for Distributed Component Simulation Environments* die Vorteile von serviceorientierten Architekturen in verteilten Simulationen dar. Der Mehrwert von MSaaS wird darin durch die gesteigerte Interoperabilität und die Wiederverwendbarkeit von Services begründet.

Simulationsservices reagieren automatisch auf einen Auftrag in der verteilten Simulation, der als Nachricht übertragen wird. Die Reaktion eines Simulationsservices wird nicht durch einen Bediener oder einen Übenden bestätigt, sodass die Antwort, je nach Berechnungszeit des Auftrags, unmittelbar erfolgen kann. Simulationsservices basieren also nicht auf menschlichen Interaktionen und werden daher auch nicht als human-in-the-loop bezeichnet (vgl. Abschnitt 2.1).

Nachfolgend sind für die drei Anwendungsfälle vor, während und nach einem Simulationsverlauf (**Pre, Inter und Post-Simulationsservice**) exemplarische Softwareprodukte für Simulationsservices aufgeführt:

- **Pre-Service:** Ein Service zur Bereitstellung von 3D-Gebäudemodellen während der Initialisierungsphase einer verteilten Simulation. Beispiel: Der *Synthetic Environment Service (SES)*. [81, 82]
- **Inter-Service:** Ein Service zur Berechnung von physikalischen Veränderungen der Umwelt während der Laufzeit der Simulation, sogenannte *Effekt-Services*. Beispiele aus dem VIntEL-Projekt (vgl. Abschnitt 2.7): *Weapon Effect Service (WES)* für Schusswaffen, *Exterior Ballistic Service (EBS)* für Mörsergeschosse, der *LFK-Dienst* für Lenkflugkörper. [26, 38, 52, 63, 91]
- **Post-Service:** Ein Service zur Aufzeichnung und Bereitstellung von Statistiken ei-

nes Simulationsverlaufs sowie zur nachträglichen Betrachtung und Verifikation der ausgetauschten Daten. Beispiel aus dem VIntEL-Projekt (vgl. Abschnitt 2.7): *Federation Agreements Conformance Test Service (FACTS)* und *Federation Integration and Experimentation Rehearsal Surrogate (FIERS)*. [70]

2.4.7 High Level Architecture

Die *High-Level-Architecture (HLA)* ist eine vom *Department of Defense (DoD)* beauftragte und vom *Defense Modeling and Simulation Office (DMSO)* entwickelte Architektur für verteilte Simulationsexperimente. Sie wird seit dem Jahr 2000 unter einer Reihe von Normen [32, 33, 34, 35, 36] geführt und ist nach Pawlaszczyk [62] der derzeit aktuelle Standard für verteilte Simulationen.

„Die HLA basiert auf der Idee, eine Gesamtsimulation in mehrere einzelne, kleine Simulationen aufzuteilen, die untereinander ihre Informationen austauschen. Die Kommunikation geschieht über ein Computernetzwerk. Verwaltet werden die einzelnen Simulationen dabei von einer zentralen Komponente, der sogenannten *Run-Time-Infrastructure (RTI)*. Diese überwacht den Simulationsablauf und verwaltet die Verteilung der Daten zwischen den Einzelsimulationen“ [95].

Für jede HLA Föderation wird ein Objektmodell, das *Federation Object Model (FOM)*, definiert. In diesem Objektmodell werden alle Klassen spezifiziert, die während einer Simulation ausgetauscht werden. Aufbau und Syntax eines FOM sind im HLA-Standard definiert.

Im FOM einer HLA wird grundsätzlich zwischen Objekten und Interaktionen unterschieden. HLA-Objekte werden durch die Objektklassen des FOM und HLA-Interaktionen durch die Interaktionsklassen des FOM definiert. Ein HLA-Objekt ist eine Datenstruktur, deren Daten (Attribute) an die teilnehmenden Simulationssysteme verteilt werden und aktualisiert werden können [95]. Ein HLA-Objekt nimmt zu meist über eine längere Zeit an der Simulation teil, beispielsweise ein Fahrzeug. Je nach Art des Objekts werden die Attribute des Objekts fortlaufend aktualisiert, beispielsweise die Position eines fahrenden Fahrzeugs. Eine HLA-Interaktion ist mit einem Ereignis gleichzusetzen, das als Nachricht einmalig in der Simulation verteilt wird, beispielsweise der Abschuss einer Waffe.

Zu Beginn einer Simulation melden sich die Simulationssysteme über die RTI für eine Föderation an und teilen mit, welche Bestandteile des FOM sie subskribieren (sie aus der Simulation erhalten möchten) und welche Bestandteile des FOM sie publizieren (sie in die Simulation einbringen werden). Während der Simulation wird jede Veränderung von der RTI kanalisiert kommuniziert. Veränderungen können beispielsweise die Instanziierung eines Objekts einer Klasse, das Aktualisieren von Attributen einer Instanz oder auch das Auslösen einer Interaktion sein.

Zusätzlich wird in der HLA Spezifikation die Einhaltung der Synchronität von Föderaten durch das sogenannte Zeitmanagement festgelegt. Dabei wird zwischen Föderaten, die passiv von der Simulationszeit abhängig sind, und Föderaten, die aktiv die Simulationszeit

beeinflussen können, unterschieden. Das Zeitmanagement der HLA kann nach Wytzisk [96] als Alleinstellungsmerkmal angesehen werden.

Für weitere Erläuterungen des HLA-Standards sei an dieser Stelle auf die einschlägige Fachliteratur verwiesen. Weiterführende Informationen finden sich beispielsweise in Pawlaszczyk [62] und eine ausführliche Dokumentation des HLA-Standards in Tolk [88].

2.4.8 Distributed Interactive Simulation

Neben der HLA existieren weitere Architekturen zur Durchführung von verteilten Simulation mit militärischen Simulationssystemen. Ein Beispiel für eine alternative Architektur ist der Standard der *Distributed Interactive Simulation (DIS)* [18]. DIS gilt als technischer Vorgänger der HLA [95]. Die Kommunikation zwischen per DIS vernetzten Simulationssystemen erfolgt direkt und nicht über eine zentrale Komponente, wie die RTI in einer HLA Simulation. Der Datenaustausch der DIS wird über vordefinierte Informationsblöcke, sogenannte *Protocol Data Units (PDU)* strukturiert und erfolgt „Paket-orientiert“ über die Netzwerkprotokolle UDP und TCP. Die eigentlichen Daten sind binär codiert“ [95]. Im Unterschied zur HLA existiert kein flexibles Objektmodell, wie das FOM.

2.4.9 Real-time Platformlevel Reference FOM

Das *Real-time Platformlevel Reference Federation Object Model (RPR-FOM)* [66] ist ein von der SISO⁶ standardisiertes Objektmodell für die Beschreibung von militärischen Einheiten und deren Interaktionen während einer verteilten Simulation. Die enthaltenen Klassen des Objektmodells werden als FOM (auf Basis der Normenreihe IEEE 1516 - HLA) und auch in der Form von PDU (auf Basis der Normenreihe IEEE 1278 - DIS) definiert. Das RPR-FOM kann daher von DIS basierten und HLA basierten verteilten Simulation verwendet werden. „Das RPR-FOM wird häufig als Schnittstelle oder gemeinsamer Nenner verschiedener verteilter Simulationen verwendet“ (frei übersetzt aus [93]). Das RPR-FOM ist ein Objektmodell für HLA und DIS. Es ist eine hierarchische Auflistung verschiedenster Typen von militärischen Einheiten, deren Attributen sowie möglicher Interaktionen, die während eines Simulationsexperimentes auftreten können.

Das RPR-FOM beinhaltet die Möglichkeit Waffenwirkungen in einer verteilten Simulation parametrisiert auszutauschen. So kann per RPR-FOM beispielsweise die Objekt-Klasse eines Einschusskraters (original: „*Crater Object Class*“) mit den beschreibenden Attributen des Kraters oder eine Interaktionsklasse zur Beschreibung von Munitionsdetonationsereignissen (original: „*Munition Detonation Interaction Class*“) definiert werden.

Ein Beispiel zur Erzeugung ein Kraterobjekts: Ein Krater wird vom RPR-FOM als HLA Objekt durch die Parameter Position, Orientierung und Durchmesser definiert (original: „*Location, Orientation, CraterSize*“ [66]). Wenn während einer Simulation durch ein Simulationssystem ein Kraterobjekt entsteht, wird eine Instanz der Klasse Krater erzeugt.

⁶Die *Simulation Interoperability Standards Organization (SISO)* ist eine internationale Organisation, die sich der Förderung und der Modellierung von Interoperabilität und Wiederverwendung in der Simulation widmet [76].

Durch die RTI wird allen Simulationssystemen, die Instanzen der Klasse Krater abonniert (subskribiert) haben, die neue Kraterinstanz mit den Werten der Position, der Orientierung und des Durchmessers übermittelt. Auf Basis dieser Werte werden die Darstellungen in den Sichtsystemen individuell vom jeweiligen Simulationssystem erzeugt. Der Krater wird in allen Simulationssystemen dargestellt.

Die Darstellung des Kraters kann in den Sichtsystemen der vernetzten Simulationssysteme jedoch unterschiedlich sein. Die konkrete Geometrie eines Kraters und dessen Einbindung in das 3D-Geländemodell ist nicht Teil des RPR-FOM. Das RPR-FOM beinhaltet keine allgemeine Definition für Geometrie. Es ist nicht möglich, komplette 3D-Gebäudemodelle per RPR-FOM über die HLA zu transportieren.

Ein Beispiel zur Verwendung der Interaktionsklasse *Munition Detonation Interaction Class* ist in Abschnitt 2.7.3 dargestellt.

2.4.10 Weitere Architekturen

Für weitere und gegebenenfalls nicht standardisierte Architekturen der verteilten Simulation, wie die *Test and Training Enabling Architecture (TENA)* [84], sei auf die einschlägigen Standardwerke wie Tolk et al. [88] verwiesen.

2.5 Techniken und Verfahren zur Simulation von Waffenwirkungen

In militärischen, virtuellen Simulationen benötigen die Übenden ein Feedback des Waffeneinsatzes. Zur Berechnung und Simulation von Waffenwirkungen existieren verschiedene Methoden und Verfahren, die in diesem Abschnitt erläutert werden.

2.5.1 Klassifikation von Waffeneinsätzen gegen Ziele

Beim allgemeinen Einsatz von Waffen gegen Ziele wird zwischen drei Arten des Waffeneinsatzes unterschieden (vgl. [27, 88, 95]). Die drei Arten sind in Abbildung 2.7 dargestellt und werden im Folgenden erläutert.

- **Direktes Feuer** (original: direct fire): Als direktes Feuer wird der Beschuss eines Ziels bezeichnet, wenn das Ziel unmittelbar sichtbar ist und auf das Ziel in direkter Sichtlinie gezielt wird. Beispiele für direktes Feuer sind Handfeuerwaffen wie Gewehre oder Pistolen.
- **Indirektes Feuer** (original: indirect fire): Als indirektes Feuer wird der Beschuss eines Zielgebietes bezeichnet, wenn nicht in direkter Sichtlinie gezielt wird. Beispiele für indirektes Feuer sind Artillerie oder Granaten. In der Regel ist das Ziel nicht unmittelbar sichtbar, so dass nicht auf ein Objekt gezielt, sondern ein Zielgebiet beschossen wird.

- **Präzisionsgelenktes Feuer** (original: smart weapons): Nach Tolk et al. [88] handelt es sich bei präzisionsgelenktem Feuer um eine Kombination aus direktem und indirektem Feuer. Präzisionsgelenkte Waffen werden zu nächst, im Sinne des indirekten Feuers, in ein Zielgebiet gebracht, um dann im Sinne des direkten Feuers ein unmittelbar sichtbares Ziel zu bekämpfen. Ein Beispiele für präzisionsgelenktes Feuer sind Marschflugkörper.

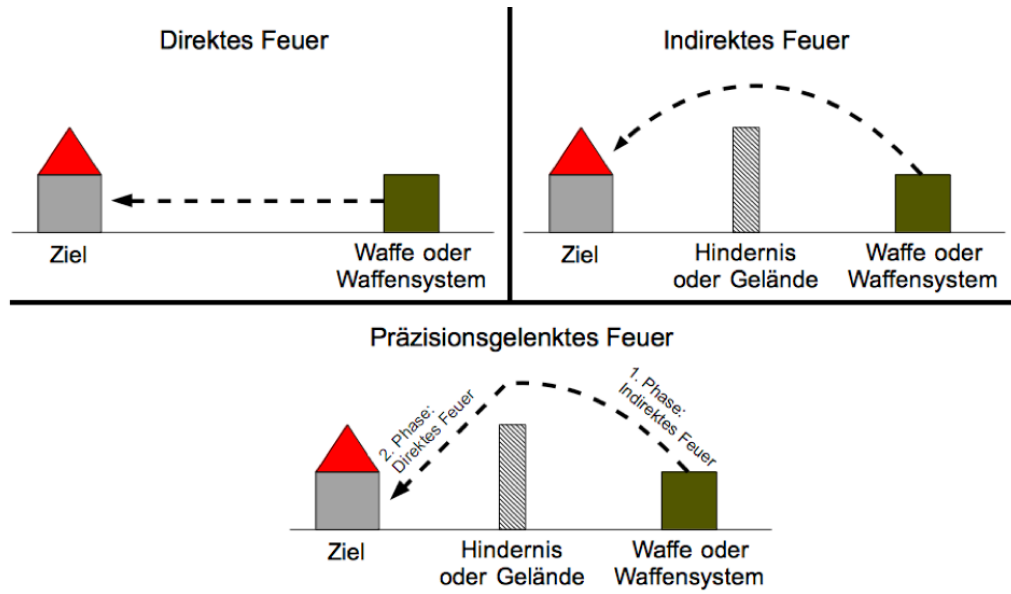


Abbildung 2.7: Klassifikation von Waffeneinsätzen.

2.5.2 Waffenwirkungen

Die Veränderung oder Beeinträchtigung eines Ziels (Beispielsweise einer Person, eines Gebäudes, eines Fahrzeugs etc.) durch Waffeneinsatz wird als *Waffenwirkung* bezeichnet. Die Waffenwirkung ist kein exakt vorhersehbares Phänomen. Welche exakten Veränderungen oder Beeinträchtigungen des Ziels durch einen Waffeneinsatz verursacht werden, ist von Einflüssen abhängig, die in der Regel durch Wahrscheinlichkeiten beschrieben werden.

Bei direktem Feuer ist ein Einschussloch im Ziel eine mögliche Wirkung des Waffeneinsatzes. Die Einschussposition eines Projektils ist jedoch nur im Idealfall an exakt der Stelle, auf die der Schütze gezielt hat. Dieser „Idealfall wäre, wenn Visierlinie, Laufseele und Trefferlage eine Einheit bilden würden. Dies wird jedoch durch äußere Umstände wie die durch die Gravitation gebogene Schussbahn, Wettereinflüsse (Wind, Regen) oder Zielfehler bei gleißendem oder schwindendem Licht verhindert“ [95]. Durch die Unsicherheiten dieser äußeren Einflüsse ist es üblich die Waffenwirkungen, beispielsweise die Position des Einschusslochs, durch Wahrscheinlichkeiten darzustellen.

Ein Beispiel einer Wahrscheinlichkeit der Waffenwirkung ist die sogenannte Treffergenauigkeit. Die Treffergenauigkeit ist die Wahrscheinlichkeit, dass bei wiederholendem Beschuss eine bestimmte Anzahl der Treffer innerhalb einer Fläche liegt. Ein konkrete Treffergenau-

igkeit ist der sogenannte *Circular Error Propable (CEP)* (deutsch: Kreisfehlerwahrscheinlichkeit). „Der CEP gibt bei einer kreisförmigen Normalverteilung den Radius eines Kreises an, in dem 50 Prozent aller Messwerte“ (Treffer) liegen [95].

Auch die Waffenwirkung von indirektem oder präzisionsgelenktem Feuer ist nicht exakt vorhersehbar. Für die Zielgenauigkeit von Artilleriegeschossen ist beispielsweise die Angabe von Zielradien in Metern üblich (vgl. [17]).

2.5.3 Simulation von Waffenwirkungen

Waffenwirkungen werden durch Wahrscheinlichkeiten beschrieben. Diese Wahrscheinlichkeiten führen in Simulationen dazu, dass das Ergebnis einer Waffenwirkung in der Simulation nicht eindeutig ist. Es gibt nur wahrscheinliche Wirkungen eines Waffeneinsatzes und weniger wahrscheinliche Wirkungen eines Waffeneinsatzes. Bei der Berechnung einer Waffenwirkung in einer Simulation existieren keine richtigen oder falschen Ergebnisse, sondern wahrscheinliche oder weniger wahrscheinliche Waffenwirkungen. Im Beispiel des direkten Feuers, gibt es in der Simulation keine eindeutige korrekte Einschussposition, sondern nur ein statistisches Modell, das die Wahrscheinlichkeit eines Treffers in einer bestimmten Fläche angibt.

Für die Berechnung der Wahrscheinlichkeiten existieren kommerzielle Softwareprodukte. Diese Softwareprodukte werden als Verwundbarkeitsmodelle oder Waffenwirkungssimulationen bezeichnet. Verwundbarkeitsmodelle, wie das deutsche *Universelle Verwundbarkeitsmodell (UniVeMo)* [38], oder Software zur Waffenwirkungssimulationen, wie die schwedische Software *Assessment of Vulnerability And Lethality (AVAL)* [5], bieten die Möglichkeit, die Wahrscheinlichkeiten von Waffenwirkungen zu berechnen. Die Resultate der Berechnungen sind beispielsweise „die Wahrscheinlichkeiten des Verlustes einer Funktion, z.B. Mobilität, Feuerkraft oder Kommunikation“ [38]. „Der Zweck von AVAL ist es, die Beurteilung von Plattform-Anfälligkeiten und die Letalität von Waffen zu unterstützen“ [5] (frei übersetzt).

Die Möglichkeiten von Verwundbarkeitsmodellen und Waffenwirkungssimulationen gehen weit über die Bestimmung der Einschlagsposition bei direktem Feuer hinaus. So können durch die Nachbildung von kompletten Waffensystemen, deren Komponenten wie Motoren oder Waffen und Materialeigenschaften der Panzerung, komplexe Waffenwirkungssimulationen berechnet werden. Die Ergebnisse einer solchen Simulation sind in Abbildung 2.8 dargestellt. In der Abbildung 2.8 ist die Wahrscheinlichkeit (zwischen 0 = unmöglich und 1= garantiert) dargestellt, dass bei einem Beschuss an der jeweiligen Position des dargestellten Fahrzeugs entweder die Mobilität, die Kampfunfähigkeit oder die tödliche Verletzung mindestens dreier Insassen zu erwarten ist. Für die Simulation wurde die Beschussstärke auf der Basis unterschiedlicher, typischer unkonventioneller Sprengvorrichtungen (*Improvised Explosive Devices (IED)*) verwendet (Beschreibung der Abbildung frei übersetzt aus [5]).

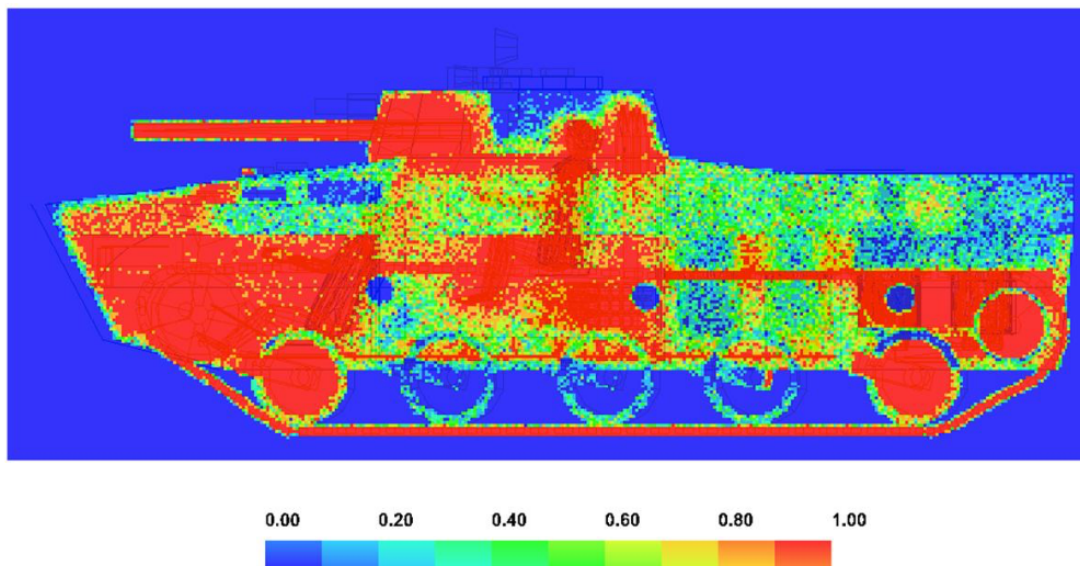


Abbildung 2.8: Beispielhafte Ergebnisse einer Verwundbarkeitssimulation aus AVAL [5]. Dargestellt ist die Wahrscheinlichkeit, dass ein panzerbrechendes Geschoss an der Einschlagsposition eine Immobilität und/oder Kampfunfähigkeit des Fahrzeugs sowie von mindestens 3 Insassen verursacht.

2.5.4 Waffenwirkungen in der virtuellen Simulation

Die Ergebnisse von Verwundbarkeitsmodellen sind zu meist statistische Modelle, beispielsweise Wahrscheinlichkeiten, möglicher Waffenwirkungen. Die Verwendung von Wahrscheinlichkeiten führen in der virtuellen Simulation zu einem Problem. In der virtuellen Simulation benötigt der Übende ein visuelles Feedback des Waffeneinsatzes und keine Wahrscheinlichkeit. Um eine einzelne Waffenwirkung im Sichtsystem darstellen zu können, wird aus dem statistischen Modell nur eine einzelne Stichprobe benötigt. Mit dieser Stichprobe muss eine konkrete Beschädigung visualisiert werden.

Ein Beispiel: In einer vernetzten Simulation mehrerer Fahrsimulationssysteme wird das Szenario einer Kolonnenfahrt im Einsatz simuliert. Das erste Fahrzeug der Kolonne wird durch ein IED angesprengt. Die Waffenwirkung der Sprengfalle entspricht der Wirkung in Abbildung 2.8. Nun wird für die beteiligten Fahrzeuge der virtuellen Simulation ein Feedback benötigt, welche Auswirkungen die Sprengfalle auf das angesprengte Fahrzeug hat. Beispielsweise benötigen die Übenden eine Darstellung des beschädigten Fahrzeugs. Die Ergebnisse eines Verwundbarkeitsmodells sind jedoch nur Wahrscheinlichkeiten eines möglichen Schadens. Wie das beschädigte Fahrzeug aussieht ist unbekannt (vgl. Abbildung 2.8.). Aus den statistischen Ergebnissen von Verwundbarkeitsmodellen können für die Anwendung in virtuellen Simulationen keine visuellen Rückkopplungen der Waffenwirkung erzeugt werden.

2.6 Techniken und Verfahren von 3D-Gebäudemodellen

Für virtuelle Simulationen werden Ausgangsdaten zur Darstellung einer virtuellen Landschaft benötigt. Beispiele für diese Ausgangsdaten sind Luftbilder, Geländemodelle und 3D-Gebäudemodelle. In dieser Arbeit werden 3D-Gebäudemodelle zur Handhabung von Gebäudeschäden betrachtet. Bekannte Techniken und Verfahren zu 3D-Gebäudemodellen werden in diesem Abschnitt dargestellt.

2.6.1 Definition und Eigenschaften

Ein 3D-Gebäudemodell wird, im Kontext dieser Arbeit, als eine dreidimensionale, virtuelle Nachbildung der äußeren Oberfläche eines realen Gebäudes definiert. Da in dieser Arbeit ausschließlich Simulationssysteme bemannter Waffensysteme betrachtet werden und diese während der Simulation zumeist außerhalb von Gebäude agieren, kann die Modellierung von Gebäudeinnenräumen vernachlässigt werden.

Ein 3D-Gebäudemodell weist unterschiedliche Eigenschaften auf. Die Geometrie, die Semantik und der Raumbezug sind Bestandteile von 3D-Gebäudemodellen die im Kontext dieser Arbeit von besonderer Bedeutung sind und die im Folgenden dargestellt werden.

Geometrie 3D-Gebäudemodelle müssen das Erscheinungsbild des realen Gebäudes wiedergeben. Dafür wird eine Repräsentation der äußeren Geometrie des Gebäudes benötigt. Grundsätzlich existieren mehrere Repräsentationsformen von Gebäudegeometrien. Vier verschiedene Repräsentationsformen sind in Tabelle 2.3 dargestellt.

Eine verbreitete Repräsentationsform ist die *Boundary Representation (BRep)*. „Die BRep, auf Deutsch Begrenzungsflächenmodell, ist eine Darstellungsform eines Flächen- oder Volumenmodells, in der Objekte durch ihre begrenzenden Oberflächen beschrieben werden“ [95]. Durch die BRep wird beispielsweise ein Würfel (Volumenmodell) durch sechs Flächen repräsentiert. Jede Fläche setzt sich wiederum aus vier Kanten zusammen, die ihrerseits aus zwei Punkten, auch Knoten genannt, bestehen.

Wenn an Stelle der Flächen des Begrenzungsflächenmodells nur die Kanten dargestellt werden, wird die Repräsentationsform auch als *Drahtgittermodell* (original: wire frame model) bezeichnet. „Ein wire frame ist ein dreidimensionales geometrisches Modell, das einen Körper lediglich durch seine Kanten repräsentiert“ [95].

Eine Alternative zur BRep ist die *Constructive Solid Geometry (CSG)* [58, 67], bei der, im Gegensatz zur Beschreibung der Begrenzung, die Volumenmodelle auf Basis von Aggregationen geometrischer Primitive definiert werden. Im CSG Modell in Tabelle 2.3 wird das 3D-Gebäudemodell auf der Basis von drei primitiven Körpern erzeugt. Der Würfel bildet das Gebäude, ein Prisma das Dach und der Quader den Schornstein des 3D-Gebäudemodells. Durch die logische Verknüpfung der Primitiven entsteht ein 3D-Gebäudemodell mit Dach und Schornstein.

Eine weitere Repräsentationsform der Geometrie ist das *Voxelmodell* (auch *Voxel Space (VS)* genannt). Ein Voxel entspricht der Form eines Würfels in einem dreidimensiona-

len Gitter. Das 3D-Gebäudemodell wird aus einer Vielzahl von Voxeln gebildet. Oberflächen des realen Gebäudes, die nicht parallel zu den Kanten der Voxel verlaufen, werden im Voxelmodell stufenartig approximiert, analog zum *Treppeneffekt* bei Digitalfotos (vgl. *Rastergrafik* aus [95]). Das Voxelmodell eines Gebäudes ist in Tabelle 2.3 rechts dargestellt. Die Anzahl der Voxel wird auch als Auflösung bezeichnet. Je mehr Voxel für das 3D-Gebäudemodell verwendet werden, also je kleinteiliger das dreidimensionale Gitter ist, desto geringer ist der Treppeneffekt.

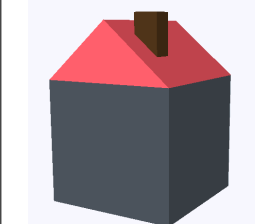
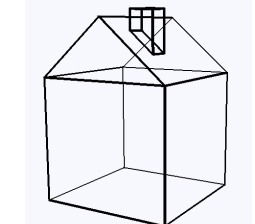
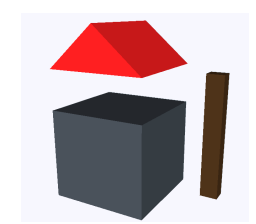
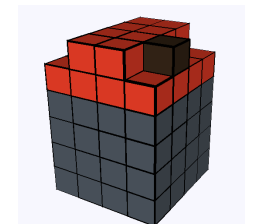
BREP Modell	Drahtgittermodell	CSG Modell	Voxelmodell
			

Tabelle 2.3: Vergleich verschiedener Geometriemodelle.

Semantik „Die Semantik bildet eine Ergänzung zur Geometrie. Während die Geometrie nach dem *-wo-* fragt, bezieht sich die Frage der Semantik nach dem *-was-*. [...] Die Semantik ist die Bedeutung eines Geoobjekts im fachspezifischen Kontext“ [95]. „Die Semantik eines Geoobjekts ist in der Regel durch Zugehörigkeit zu einer bestimmten Art/einer Klasse von Objekten gegeben“ [25].

Der fachspezifische Kontext oder die Zugehörigkeit zu einer bestimmten Art ist bei 3D-Gebäudemodellen nicht immer eindeutig und „nicht frei von Subjektivität“ [95]. Ein Beispiel: Für ein Gebäude, welches zur Unterbringung oder Reparatur von Vehikeln dient, können je nach Kontext die Bezeichnungen *Garage*, *Werft* oder *Hangar* verwendet werden. Der fachspezifische Kontext des Gebäudes richtet sich nach der Art der Vehikel, die in dem Gebäude untergestellt werden. Im maritimen Kontext ist die Bezeichnung *Werft*, in der Luftfahrt die Bezeichnung *Hangar* und für Landfahrzeuge die Bezeichnung *Garage* üblich. Die Semantik des 3D-Gebäudemodells ist vom Kontext abhängig.

Die Semantik eines 3D-Gebäudemodells ist dabei nicht nur auf einen einzelnen Begriff beschränkt, der die Funktion des Gebäudes beschreibt (*Garage*, *Werft* oder *Hangar*). Die Semantik eines 3D-Gebäudemodells kann durch komplexe hierarchische Modelle definiert werden, die neben der Funktion des Gebäudes auch die Bestandteile von Gebäuden wie *Wände*, *Dächer*, *Türen* oder *Fenster* beschreiben (vgl. Abschnitt 2.6.3).

Raumbezug Ein weiterer Bestandteil von 3D-Gebäudemodellen ist der *Raumbezug*. Der Raumbezug von 3D-Gebäudemodellen beschreibt die Position eines 3D-Gebäudemodells in einem Bezugssystem. Dies wird durch Definition und Verwendung von globalen, nationalen oder lokalen Bezugsrahmen, den *Koordinatenreferenzsystemen* (*CRS*⁷) gelöst. Durch

⁷Abkürzung aus dem Englischen *coordinate reference system*

parallele Entwicklungen in der Vergangenheit existieren heute eine Vielzahl von CRS. Maßgeblichen Einfluss auf die Beschreibung und Katalogisierung von CRS und deren Transformationen hat die *European Petroleum Survey Group Geodesy (EPSG)*. Die EPSG verwaltet und publiziert einen Katalog mit nationalen und globalen Koordinatenreferenzsystemen. Nahezu jedes CRS ist darin mit einem eindeutigen Schlüssel, dem sogenannte EPSG-Code, sowie weiteren beschreibenden Attributen und Transformationsparametern, katalogisiert.

2.6.2 Vergleich von Normen und Standards für 3D-Gebäudemodelle

Zur Speicherung, zum Austausch und zur Visualisierung von 3D-Gebäudemodellen sind in der virtuellen Simulation verschiedene Datenmodelle und Datenformate bekannt, die auf unterschiedlichen Normen und Standards basieren. Drei Beispiele für Normen und Standards werden in den folgenden drei Abschnitten 2.6.3, 2.6.4 und 2.6.5 erläutert. Eine Übersicht der Eigenschaften, auf die in den folgenden drei Abschnitten näher eingegangen wird, ist in Tabelle 2.4 dargestellt.

	CityGML	SEDRIS	Open Flight
Norm oder Standard	ISO 19107	ISO/IEC 18023	OGC CDB
Geometrische Repräsentation	BRep	BRep	BRep
Semantische Repräsentation	ja, zwingend	ja, optional	nein
Level of Detail-Methodik	semantisch	geometrisch	geometrisch
Anzahl Level of Detail	maximal 5	unbegrenzt	unbegrenzt
Zustandsrepräsentation	nein	ja, per Level of States	ja, per Level of States
Dateikodierung	Text, XML	binär	binär

Tabelle 2.4: Tabellarischer Vergleich von Normen und Standards für 3D-Gebäudemodelle in der virtuellen Simulation.

2.6.3 3D-Gebäudemodelle auf Basis der Norm ISO 19107

3D-Gebäudemodelle können für die virtuelle Simulation auf der Basis der Norm ISO 19107 [78], dem sogenannten Spatial Schema, definiert werden. Dazu werden die auf der ISO 19107 basierenden Standards GML und CityGML verwendet. Spatial Schema, GML und CityGML werden im Folgenden beschrieben.

Spatial Schema Das *Spatial Schema* ist eine Norm der *International Organization for Standardisation (ISO)*. Es wird unter der Kennung ISO 19107 [78] geführt und definiert ein generelles Schema zur Formalisierung von Geometrie und Topologie. Kerninhalt der Norm ist die Beschreibung von geometrischen und topologischen Primitiven. Dabei werden vektorgeometrische Primitive wie Punkte, Linien, Polygone und deren mögliche topologische Beziehungen auf Basis der Egenhofer-Relationen definiert (vgl. [19]).

Geography Markup Language (GML) „Die *Geography Markup Language (GML)* ist eine Spezifikation des OGC⁸ und wurde entwickelt, um einen interoperablen Datenaustausch von vektorbasierten Geodaten zwischen verteilten Systemen zu ermöglichen“ [79]. Das grundlegende Modellierungskonzept von GML unterstützt Geometrie der Boundary Representation unter der Verwendung von räumlichen Bezugssystemen nach Definition der EPSG.

Die Spezifikation beschreibt außerdem die Verknüpfung räumlicher und nicht-räumlicher Objekte durch Beziehungen auf Basis der *Unified Modeling Language (UML)*. Ein wesentlicher Bestandteil von GML ist die Adaptierbarkeit für spezielle Anwendungsgebiete. Zwar umfasst GML die Formalisierung von Geometrien, per se aber nicht deren Semantik. Mit anderen Worten ist ein Objekt in GML zwar geometrisch eindeutig beschrieben, semantische Informationen, wie zum Beispiel *Dach* oder *Haus*, fehlen jedoch. Für semantische Erweiterungen bietet GML die Möglichkeit einer Spezialisierung durch sogenannte Anwendungsschemata⁹. Ein Beispiel für ein GML Anwendungsschema, das speziell für 3D-Gebäudemodelle entwickelt wurde, ist die *City Geography Markup Language (CityGML)*.

City Geography Markup Language (CityGML) CityGML ist ein sogenanntes *Anwendungsschema* von GML für die eindeutige Beschreibung und den Austausch von Geometrie und Semantik von virtuellen 3D-Stadtmodellen. CityGML ist sowohl ein Datenmodell als auch ein Datenaustauschformat [14]. In CityGML werden Stadt- und Landschaftsobjekte, insbesondere Gelände, Gebäude, Wasser- und Verkehrsflächen, Vegetation, Stadtmöblierung und Landnutzungen modelliert. Dabei werden neben der Geometrie insbesondere auch die Semantik der Objekte beschrieben [42, 73, 95]. CityGML wird von der *Special Interest Group 3D (SIG3D)* [73] entwickelt und darüber hinaus seit 2008 als offizieller Standard des OGC geführt.

Ein wesentlicher Bestandteil von CityGML ist die Semantik basierte Beschreibung der *Level of Detail (LoD)*. Das LoD-Konzept beschreibt einen allgemeinen Ansatz zur Vorhaltung mehrerer Repräsentationsformen eines Objekts. Dabei wird beispielsweise ein 3D-Gebäudemodell parallel in unterschiedlichen Stufen definiert. Diese Repräsentationsformen werden als Detaillierungsstufen (Englisch: Level of Detail) bezeichnet. Nach Stüber [80] liegen die Schwerpunkte der in der Literatur aufgeführten Definitionen für Level of Detail „zum einen in der Beschreibung, welche Objekte beziehungsweise Objektarten in den einzelnen Stufen zur Ansicht kommen und zum anderen, nach welchen Kriterien [...] verschiedene LoD-Zonen bestimmt werden können“ [80].

CityGML beinhaltet fünf vordefinierte LoD für ein 3D-Gebäudemodell. Grafische Beispiele der unterschiedlichen Repräsentationsstufen sind in Abbildung 2.9 dargestellt. Die LoD-Stufen von CityGML kennzeichnet, dass sie neben der geometrischen Simplifizierung (vgl. Abbildung 2.11) auch die Semantik von 3D-Gebäudemodellen berücksichtigen. Ein LoD-1 3D-Gebäudemodell ist in CityGML nicht nur geometrisch gegenüber dem LoD-2 Modell

⁸Seit 2009 wird GML außerdem unter der ISO Norm 19136:2007 geführt.

⁹Die Erstellung von GML-Anwendungsschemata werden in der ISO Norm 19109 definiert.

vereinfacht, sondern auch semantisch. Während ein LoD-2 Modell auch Dachflächen enthalten kann, ist dies per Definition bei einem LoD-1 Modell nicht erlaubt. Das LoD-Konzept von CityGML wird daher im Folgenden als semantisch bezeichnet. In Tabelle 2.5 und Abbildung 2.9 sind die Unterschiede der CityGML LoD dargestellt.

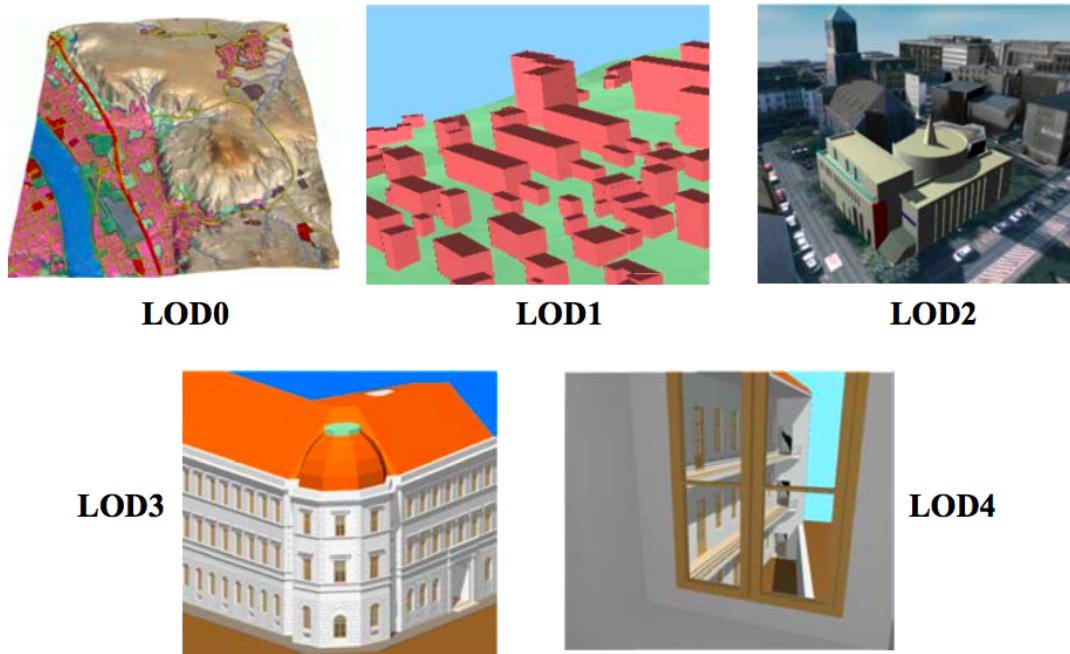


Abbildung 2.9: Level of Details von CityGML aus [14].

	LoD 0	LoD 1	LoD 2	LoD 3	LoD 4
Anwendungsbereich	Region	Stadt	Stadtteil	Einzelgebäude (außen)	Einzelgebäude (innen)
Genauigkeitsklasse	geringste	gering	normal	höher	sehr hoch
Metrische Genauigkeit	< LoD 1	5 m	2 m	0,5 m	0,2 m

Tabelle 2.5: Tabellarischer Vergleich der CityGML LoD (frei übersetzt und gekürzt aus [14, 42]).

Beispiele von CityGML in der virtuellen Simulation Die erste konzeptionelle Brücke von CityGML und Simulation wurde 2010 von Averdung [6] vorgestellt. Durch die konsequente Abbildung von CityGML in objektrelationalen Datenbanken wird die Versorgung von Simulationssystemen aus solchen Datenbanken gezeigt. Rossmann et al. [65] verwenden diesen Ansatz in verschiedensten zivilen Anwendungsgebieten der Simulation. Die Veränderung der CityGML 3D-Gebäudemodelle während der Simulation wird jedoch noch als Bestandteil zukünftiger Forschungsarbeiten dargestellt [46].

2.6.4 3D-Gebäudemodelle auf Basis der Norm ISO/IEC 18023

Die Norm ISO/IEC 18023^{10 11} beinhaltet mit der *Synthetic Environment Data Representation and Interchange Specification (SEDRIS)*¹² eine Möglichkeit zur Definition von 3D-Gebäudemodellen. „SEDRIS ist eine Infrastrukturtechnologie, die die Beschreibung, Definition, Wiederverwendung und den Austausch von Umweltdaten ermöglicht. SEDRIS bietet die Möglichkeit Umweltdaten (Gelände, Meer, Luft und Weltraum) zu repräsentieren und diese eindeutig, verlustfrei und nicht-proprietär austauschen zu können“ (frei übersetzt aus der SEDRIS-Spezifikation [69]). Die SEDRIS Spezifikationen umfassen sowohl ein Datenmodell, als auch ein Datenaustauschformat. Das Datenmodell von SEDRIS umfasst laut Spezifikation das Geschehen auf, unter und über der Erde [69].

SEDRIS wurde 1994 unter der Schirmherrschaft des DoD entwickelt und wird heute von einer globalen Anwenderschaft im Bereich der Simulation unterstützt. In SEDRIS können abstrakte Zusammenhänge, beispielsweise die Beziehung zwischen Objekten und deren Semantik, unabhängig voneinander modelliert werden. Dafür unterscheidet SEDRIS zwischen abstrakten Klassen und deren Semantik. Bestandteil von SEDRIS sind sowohl ein abstraktes Klassenmodell, das *Data Representation Model (DRM)*, als auch die Klassifikationsbibliothek *Environmental Data Coding Specification (EDCS)*. Das DRM kann als die Grammatik und der EDCS-Code als das Wörterbuch von SEDRIS gelten.

In [44] wird dargestellt, dass sich die Definition für 3D-Gebäudemodelle in SEDRIS vom Standard CityGML unterscheidet. Zwar können auch in SEDRIS 3D-Gebäudemodelle mit Geometrie und Semantik definiert werden, allerdings umfasst der Standard SEDRIS auch Phänomene, die über die Definitionen von CityGML hinaus gehen. Diese Unterschiede sind plakativ in Abbildung 2.10 dargestellt.

SEDRIS beinhaltet, ebenso wie CityGML, ein LoD-Konzept zur Definition unterschiedlicher Repräsentationsstufen. Im Unterschied zu CityGML, basiert die Generalisierung zwischen den LoD nur auf der geometrischen Simplifizierung. Die verschiedenen LoD dienen der Sichtdistanz abhängigen Reduktion von Dreiecken (vgl. Abbildung 2.11) [95].

2.6.5 3D-Gebäudemodelle auf Basis des OGC Common Database

Neben CityGML und SEDRIS wird in der virtuellen Simulation zur Definition von 3D-Gebäudemodellen eine Kombination der Standards CDB und Open Flight verwendet. Die *Common Database (CDB)* ist eine Spezifikation für synthetische Umweltdatenbanken. CDB ist seit 2016 ein Standard des OGC [13]. CDB wurde von *Canadian Aviation Electronics (CAE)*, einem Hersteller von Flugsimulatoren, entwickelt und wird unter anderem in den Produkten der Firma Presagis verwendet. „Die CDB ist ein offenes Format

¹⁰SEDRIS umfasst neben der funktionalen Spezifikation in der ISO/IEC 18023 auch die Normen ISO/IEC 18024-4:2006, ISO/IEC 18025:2005, ISO/IEC 18026:2006, ISO/IEC 18041-4:2005 und ISO/IEC 18042-4:2006.

¹¹„Die *International Electrotechnical Commission (IEC)* [...] ist eine internationale Normungsorganisation für Normen im Bereich der Elektrotechnik und Elektronik [...]. Einige Normen werden gemeinsam mit der ISO entwickelt“ [95].

¹²frei übersetzt: Spezifikation zur Repräsentation und zum Austausch von synthetischen Umweltdaten.

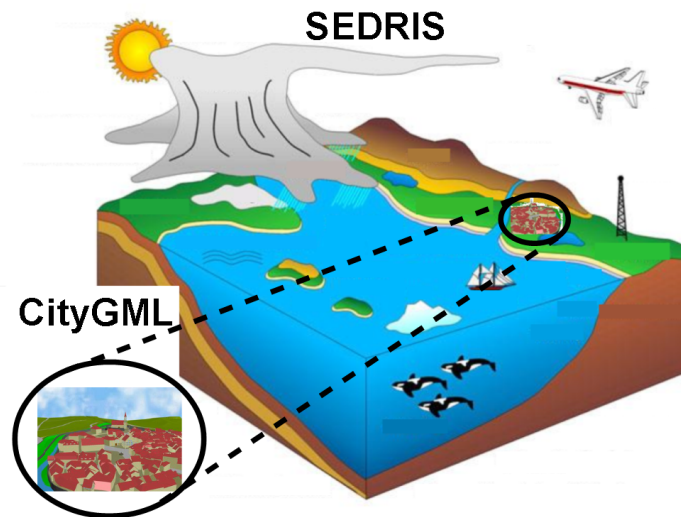


Abbildung 2.10: Vergleich von SEDRIS und CityGML (CityGML Abbildung aus [14], SEDRIS Abbildung aus [69])

für die Speicherung, den Zugriff und die Änderung einer synthetischen Umweltdatenbank. Die Spezifikation definiert die Datendarstellung, die Organisations- und Speicherstruktur einer weltweiten synthetischen Darstellung der Erde. [...] CDB umfasst Luftbilder, 3D-Geländemodelle und (3D-) Modelle von natürlichen und anthropogenen Strukturen der Erde“ [13] (frei übersetzt). CDB bietet keine direkte Möglichkeit 3D-Gebäudemodelle zu definieren, sondern verweist intern auf das Datenformat Open Flight [61].

Open Flight wurde 1988 vom Unternehmen *Software Systems Incorporated* veröffentlicht. Es gilt heute als Quasistandard zum Austausch von 3D-Daten in der virtuellen Simulation. Open Flight ist sowohl ein Datenmodell, wie auch ein Datenformat. Der methodische Ansatz von Open Flight ist die Codierung von Szenengraphen (vgl. Abschnitt 2.6.6) in Binärdateien zum effizienten Austausch einer räumlichen Szene. Durch die direkte Abbildung der Szenengraphen in Open Flight-Dateien wird der Transformationsaufwand, also die Codierung oder Decodierung von Geometrie, zwischen Systemen minimiert. Open Flight beschreibt ausschließlich die Geometrie der 3D-Objekte und keine Semantik.

In der aktuellsten Version 16.4 von Open Flight können auch sogenannte *Switch-Knoten* des Szenengraphen modelliert werden. Ein Switch-Knoten verweist als Elternknoten auf mehrere Kindknoten. Die Sichtbarkeit der Kindknoten ist in der Regel orthogonal. Das heißt, dass immer nur einer der Kindknoten sichtbar ist, während alle anderen deaktiviert, also nicht sichtbar sind. Die Sichtbarkeit der Kindknoten kann zur Laufzeit einer Simulation im Switch-Knoten umgeschaltet werden. Durch diesen Mechanismus wird das LoD-Konzept umgesetzt. Das LoD-Konzept von Open Flight basiert ebenso wie das LoD-Konzept von SEDRIS nur auf der geometrischen Simplifizierung.

Im Unterschied zu CityGML ist Open Flight ein Binär- und kein Klartextformat. Open Flight lässt sich also nicht wie XML in einem Browser oder Texteditor manuell bearbeiten und ist somit rein maschinell interpretierbar (vgl. *Binärformat* und *XML* aus [95]).



Abbildung 2.11: Geometrische Level of Detail zur Simplifizierung auf Basis der Anzahl der Dreiecke aus [95].

Die Common Database wird in Simulationssystemen der CAE, beispielsweise bei der Bundeswehr für den Transporthubschrauber NH90, eingesetzt (vgl. [11]).

2.6.6 Visualisierung von 3D-Gebäudemodellen

3D-Gebäudemodelle müssen im Sichtsystem visualisiert werden. Dafür werden die Daten der 3D-Gebäudemodelle in den entsprechenden Datenformaten gespeichert. Die Visualisierung der 3D-Gebäudemodelle ist die Überführung der Daten zu einer grafischen Darstellung des 3D-Gebäudemodells im Sichtsystem.

Technischer Visualisierungsprozess Die Visualisierung der 3D-Gebäudemodelle ist ein zweistufiger Prozess. Zuerst erfolgt das Laden der benötigten 3D-Gebäudemodelle von einem Permanentenspeicher (Festplatte oder Datenbank) in den Hauptspeicher des Simulationsrechners. Anschließend werden die Daten in einem zweiten Schritt in den Grafikspeicher der Grafikkarte überführt. Dabei gilt der Grundsatz *Permanentenspeicher* > *Hauptspeicher* > *Grafikspeicher*. In Abhängigkeit zur räumlichen Größe und Detailreichtum der virtuellen 3D-Landschaft ist es nicht möglich alle 3D-Gebäudemodelle gleichzeitig im Grafikspeicher vorzuhalten. Während der Simulation, also beispielsweise während des virtuellen Fliegens über ein Gelände, müssen in Abhängigkeit zur virtuellen Position des Flug- oder Fahrzeugs, neue 3D-Daten hinzugefügt beziehungsweise wieder aus dem Grafikspeicher entfernt werden. Es findet ein permanenter Datenaustausch zwischen Permanentenspeicher, Hauptspeicher und Grafikspeicher statt.

Für die Datenübertragung zwischen Permanent-, Haupt- und Grafikspeicher werden sogenannte 3D-Engines verwendet. Eine 3D-Engine ist ein Computerprogramm, das die Kommunikationsprachen von Grafikkarten (bspw. *OpenGL* oder *DirectX*) implementiert. Um die visualisierten 3D-Daten im virtuellen Raum zu positionieren, zu orientieren oder gegebenenfalls ein- und auszublenden, verwenden 3D-Engines unterschiedliche verwaltende Da-

tenstrukturen. Die wohl bekannteste Datenstruktur ist der sogenannte *3D-Szenengraph*¹³. In einem Szenengraph sind alle geometrischen Objekte in einer baumartigen Struktur räumlich organisiert, wobei jeder Knoten des Baumes eine eigene Position und Orientierung verwenden kann. Ein 3D-Szenengraph ermöglicht durch die räumliche Struktur der Szene das selbstständige Nachladen von benötigten oder das Ausblenden von nicht mehr verwendeten 3D-Gebäudemodellen, um den Grafikspeicher optimal auszunutzen. Ein Beispiel: Entfernt sich ein Flugzeug in einer virtuellen Simulation von einem Gebäude so weit, dass die Größe auf dem Bildschirm marginal wird, so wird dieses 3D-Gebäudemodell aus dem Grafikspeicher entfernt. Nähert sich das Flugzeug dagegen einem 3D-Gebäudemodell bis auf eine sichtbare Distanz, wird das 3D-Gebäudemodell entsprechend nachgeladen und zur Anzeige gebracht. Alternativ zum harten Ein- und Ausblenden wird das LoD-Konzept zur Verschaltung mehrere Detaillierungsstufen eingesetzt (vgl. Abbildung 2.11).

Geotypische und Geospezifische Visualisierung Unabhängig von Datenformat, Datenmodell und technischem Visualisierungsprozess wird bei der Visualisierung von 3D-Gebäudemodellen zwischen *geotypischen* und *geospezifischen* 3D-Gebäudemodellen unterschieden. Ein geotypisches Gebäudemodell ist kein Abbild eines realen Bauwerks, sondern es entspricht in Größe, Aussehen und Beschaffenheit in etwa den realen Gebäuden an diesem Ort. Geospezifische Gebäude sind hingegen Nachbildungen von real existierenden Bauwerken.

Im Sichtsystem eines Simulators werden geotypische und geospezifische Gebäudemodelle verwendet. Markante Bauwerke, sogenannte *Landmarks*, die durch ihre Größe und Position hervorstechen, sind zumeist geospezifische 3D-Gebäudemodelle. Ein Beispiel für ein geospezifisches 3D-Gebäudemodell ist der Seattle Space Needle der Flugsimulationssoftware X-Plane [97] in Abbildung 2.12 rechts. Das Gebäude ist durch seine Markantheit und Bekanntheit eine wichtige Orientierungshilfe in der Navigation und daher geospezifisch modelliert. Ähnliche Gebäude werden dagegen als geotypische 3D-Gebäudemodelle wiederholt dargestellt. So ist in Abbildung 2.12 links nicht jedes Reihenhaus einzeln modelliert, sondern ein für diese Region typisches Reihenhaus wird mehrfach an unterschiedlichen Positionen dargestellt. Viele Häuser sind identisch.

Eine rein geospezifische Darstellung aller 3D-Gebäudemodelle ist zwar theoretisch möglich, wäre aber in der Erfassung der Daten mit enormem Aufwand verbunden. Da geotypische 3D-Gebäudemodelle durch die wiederholte Darstellung nur einmalig Grafikspeicher reservieren, reduzieren sie grundsätzlich die Speicherlast und begünstigen hohe Bildwiederholraten.

¹³„Ein Szenengraph ist eine Datenstruktur, die häufig bei der Entwicklung computergrafischer Anwendungen eingesetzt wird. Es handelt sich um eine objektorientierte Datenstruktur, mit der die logische, in vielen Fällen auch die räumliche Anordnung der darzustellenden zwei- oder dreidimensionalen Szene beschrieben wird“ [95].



Abbildung 2.12: Darstellung von geotypischen (links) und geospezifischen 3D-Gebäudemodellen (rechts) aus der Flugsimulationssoftware X-Plane [97].

2.7 Stand der Technik - Handhabung von Gebäudeschäden

In diesem Abschnitt wird der aktuelle Stand der Technik zur Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation am Beispiel des Bundeswehr Forschungsvorhabens *Systemdemonstrator Verteilte, Integrierte Erprobungslandschaft (VIntEL)* dargestellt.

2.7.1 Kurzbeschreibung VIntEL

Nach aktuellem Kenntnisstand existiert nur ein Projekt, welches Gebäudeschäden in Joint Exercises während der virtuellen Simulation thematisiert. Dies ist das Forschungsprojekt *Systemdemonstrator Verteilte, Integrierte Erprobungslandschaft (SD VIntEL)*. VIntEL war ein Forschungsprojekt des *Bundesamtes für Ausrüstung, Informationstechnik und Nutzung der Bundeswehr (BAAINBw)* zwischen den Jahren 2005 und 2015. „Der Auftrag des Vorhabens SD VIntEL ist [...] der Aufbau einer verteilten integrierten Erprobungslandschaft als Systemdemonstrator mit dem Ziel, mit Hilfe dieses Testbeds frühzeitig Aufschluss über die wehrtechnische Relevanz innovativer Ideen und technologischer Konzepte zur NetOpFü¹⁴ im realitätsnahen operativen Einsatz zu erhalten. In der Umsetzung [...] ist hierzu die Kopplung vorhandener Simulations- und Realsysteme mit dem Ziel durchzuführen, eine erste Bewertung von neuen – teils noch nicht realisierten – Waffensystemen, technologischen Konzepten und Ideen zu ermöglichen.“ [26]

Die im Jahr 2013 an einem Simulationsexperiment des Projekts beteiligten Simulationssysteme und Services sind in Abbildung 2.13¹⁵ dargestellt. Beteiligte Institutionen des VIntEL Projekts sind neben dem BAAIN Bw und der Universität der Bundeswehr München auch diverse Konzerne und Unternehmen der Rüstungsindustrie oder des Geoinformationssektors. Ohne Anspruch auf Vollständigkeit sind dabei zu nennen: *Rheinmetall*

¹⁴ *Vernetzte Operationsführung (NetOpFü)* vgl. *Network-Centric Warfare* aus [95].

¹⁵ Für die Definition aller verwendeten Akronyme in der Abbildung sei auf [26, 51] verwiesen.

Defence Electronics (RDE) [64], Industrieanlagen-Betriebsgesellschaft mbH (IABG) [38], Thales Group [85], Cassidian [12] und CPA ReDev [15].

Ein Ergebnis des Projekts ist die Entwicklung eines gemeinsamen Datenbankschemas zur Bereitstellung einer für alle Simulationssysteme identischen, virtuellen 3D-Landschaft zur Steigerung von Interoperabilität und Fair-Fight. Im Datenbankschema ist auch die Definition von Zustandsstufen der 3D-Gebäudemodelle für Gebäudeschäden definiert. Das gemeinsame Datenbankschema wird auf Basis von GML und SEDRIS (vgl. Abschnitt 2.6) entwickelt und unter anderem von Stüber und Krückhans [45, 82] publiziert. Konform zu diesem Datenbankschema wird eine Datenbasis verwendet, die als Pre-Service in die VIntEL-Architektur integriert wird (vgl. Abbildung 2.13). Der Pre-Service ist der *Synthetic Environment Service (SES)*.

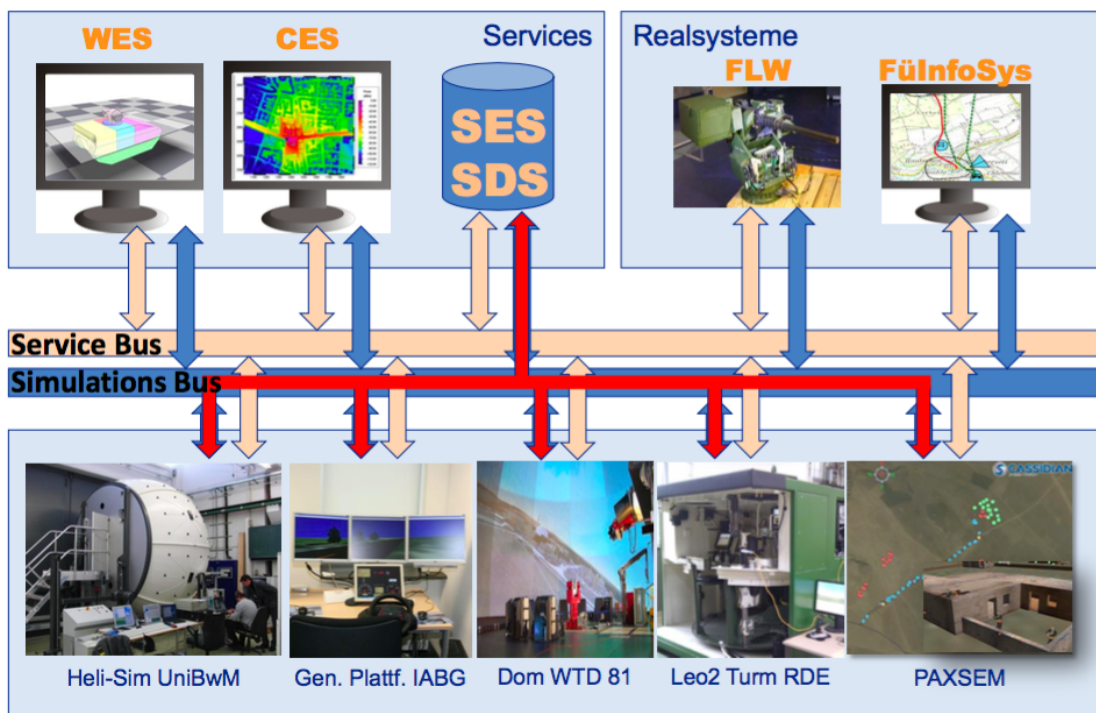


Abbildung 2.13: VIntEL Gesamtarchitektur aus [51] (Ausschnitt mit farblicher Anpassung ohne inhaltliche Änderung).

2.7.2 Gebäudeschäden vor der Simulation

Die 3D-Gebäudemodelle werden im SES unter Verwendung des kombinierten SEDRIS-GML-Schemas vorgehalten. Die Zustandsstufen, auch Level of States genannt, werden in SEDRIS-GML modelliert.

In Abbildung 2.14 wird ein Beispiel gezeigt, wie die Handhabung von Gebäudeschäden in VIntEL in SEDRIS-GML gespeichert wird. Die Abbildung 2.14 wird im Folgenden näher erläutert: Die Klasse *DRM State Related Geometry* ist ein Bestandteil des originären SEDRIS Datenmodells und repräsentiert in diesem Beispiel ein Gebäude mit mehreren vorgerechneten *zustandsbezogenen* (im Englischen: „state related“) Instanzen, also Zerstö-

rungsgraden. Jedes *DRM Union of Geometry Hierarchy* steht für eine Instanz des 3D-Gebäudemodells. Die verschiedenen *DRM Union of Geometry Hierarchy* repräsentieren die Schäden *Healthy* - keine Schäden, *Slight Damage* - leichte Beschädigung, *Medium Damage* - mittlere Beschädigung (in Abbildung 2.14 nicht dargestellt) und *Heavy Damage* - sehr starke Beschädigung. Jedem *DRM Union of Geometry Hierarchy* ist genau ein *DRM State Data* zugeordnet. Mit der *DRM State Data* wird jeder *DRM Union of Geometry Hierarchy* ein Intervall des prozentualen Beschädigungsgrades zugewiesen. Beispielsweise ist die Zustandsstufe *Slight Damage* für einen prozentualen Schaden von 0,25 bis 0,5 zu verwenden. Diese Intervallgrenzen werden im *DRM State Data* durch die Attribute *active_state.real_value.lower_bound* beziehungsweise *_upper_bound* definiert. Der aktuelle Zustand des 3D-Gebäudemodells wird im *DRM Union of Geometry Hierarchy* im Attribut *active_state.real_value.single_value* repräsentiert. Der Wert „0“ für dieses Attribut bedeutet also, dass die Beschädigung 0 Prozent ist. Entsprechend der Intervalle würde nun das unbeschädigte *healthy* 3D-Gebäudemodell visualisiert werden.

Die eigentliche Geometrie des 3D-Gebäudemodells beziehungsweise der zugeordneten Zustandsstufen ist in der Abbildung nicht dargestellt. Die Geometrie würde hierarchisch unterhalb der *DRM Union of Geometry Hierarchy* definiert werden (vgl. [47]).

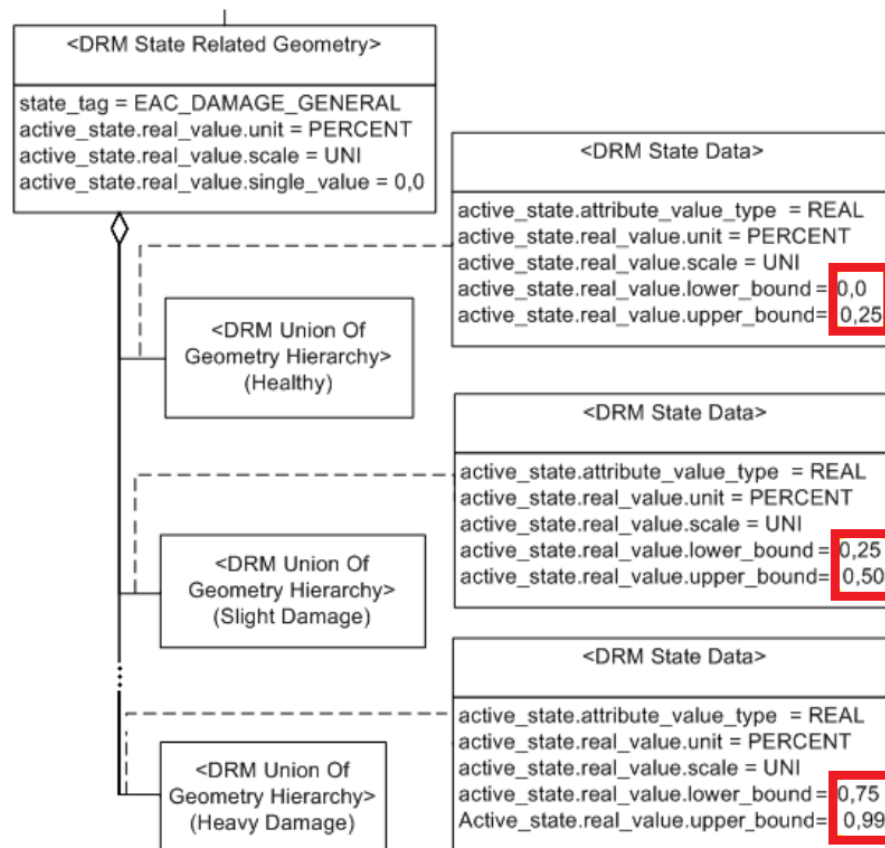


Abbildung 2.14: Attribute vorgerechneter Zustandsstufen eines 3D-Gebäudemodells aus [69, 82].

Parametername	Beschreibung (frei übersetzt)
EventIdentifier	Identifikator der Interaction.
FiringObjectIdentifier	Identifikator des feuernden Waffensystems (Simulationssystem).
DetonationLocation	Die Position, in Weltkoordinaten (globaler Raumbezug), an denen die Munition explodierte.
FinalVelocityVector	Der Geschwindigkeitsvektor der Munition im Augenblick der Detonation.
MunitionType	Art, Staat, Domäne, Kategorie und Unterkategorie [...] der detonierten Munition.
<i>RelativeDetonationLocation</i>	<i>Optional Parameter: Position relativ zur Position des Ziels an der die Munition explodierte.</i>
<i>DetonationResultCode</i>	<i>Optional Parameter: Art der Detonation.</i>
<i>RateOfFire</i>	<i>Optional Parameter: Feuerrate in Schüssen pro Minute.</i>

Tabelle 2.6: Auswahl von Parametern einer Munition Detonation Interaction in einer HLA oder DIS Simulation des RPR-FOM Standards (Ausschnitt frei übersetzt aus [66]).

2.7.3 Gebäudeschäden während der Simulation

Während der Simulation ändert sich ausschließlich der prozentuale Beschädigungsgrad auf attributiver Ebene. Die Änderung des Beschädigungsgrades führt so dann zur Umschaltung der dargestellten Schadensstufe. Zwischen den Simulationssystemen wird während der Simulation keine Geometrie ausgetauscht, sondern nur der veränderte Zustand durch die Angabe der prozentualen Zerstörungsgrades propagiert. Das Sichtsystem aktualisiert daraufhin die Darstellung des Gebäudes, indem die alte Zustandsstufe aus- und die neue Zustandsstufe eingeblendet wird.

Interaktionen des RPR-FOM Innerhalb des RPR-FOM kann die Ursache eines Gebäudeschadens beispielsweise als Interaktion über die *Munition Detonation Interaction* übermittelt werden. „Die Munition Detonation Interaction alarmiert alle partizipierenden Simulationsteilnehmer, über die Detonation einer Waffe“ (frei übersetzt aus [66]). Andere Interaktionen oder Objekte des RPR-FOM, wie die *Crater Object Class*, definieren unmittelbare Folgen eines Waffeneinsatzes, parametrisieren jedoch keine Ursache. So beschränkt sich die Erläuterung zur *Crater Object Class* auf die Beschreibung: „Die *Crater Object Class* ist ein punkthaftes Objekt, das den Durchmesser eines Krater beschreibt“ (frei übersetzt aus [66]). Mit Ausnahme des Durchmessers enthält das Kraterobjekt weder geometrische Informationen noch Angaben zum ursächlichen Waffeneinsatz.

Die Parameter der *Munition Detonation Interaction* sind in Tabelle 2.6 dargestellt. Neben dem Ort der Detonation und dem Typ der Munition sind auch der Geschwindigkeitsvektor der Munition, also die Richtung aus der das Geschoss, die Rakete oder das Projektil einschlägt, vorhanden.

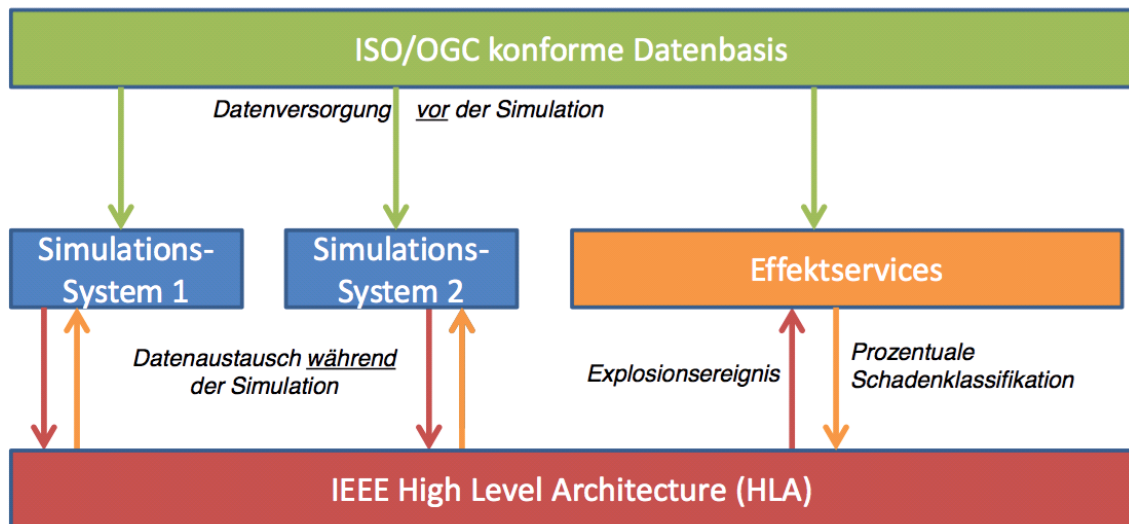


Abbildung 2.15: Prozessmodell der Handhabung von Gebäudeschäden in der Simulationsarchitektur des Projekts VIntEL.

Simulationsereignis und Schadensservice Die Kommunikation während der Simulation wird durch die HLA durchgeführt. In der HLA wird zwischen Objekten und Interaktionen unterschieden, die im Objektmodell der Simulation (Federation Object Model (FOM)) definiert werden (vgl. Abschnitt 2.4.7). Der Datenaustausch im VIntEL Projekt basiert auf dem RPR-FOM [63]. Explosionsereignisse (*Munition Detonation Interaction* des RPR-FOM) während der Simulation werden von den sogenannten Effektservices abonniert und weiterverarbeitet. Die Effektservices reagieren auf Explosionsereignisse und berechnen auf Basis der Parameter des Ereignisses und den 3D-Gebäudemodellen (die ihnen aus dem SES zur Verfügung stehen) einen prozentualen Schaden. Dieser prozentuale Schaden wird wiederum über die HLA publiziert und führt gegebenenfalls bei den beteiligten Simulationssystemen zum Umschalten der Schadensstufen der 3D-Gebäudemodelle. Ein Prozessmodell der Handhabung von Gebäudeschäden innerhalb des VIntEL-Projekts ist zusammenfassend in Abbildung 2.15 dargestellt.

In VIntEL werden nach Glose [26], in Abhängigkeit des Waffentyps, unterschiedliche Kombinationen von Schadens- oder Effektservices verwendet. Konkret sind der *Weapon Effect Service (WES)* für Schusswaffen, der unten exemplarisch beschrieben ist, der *Exterior Ballistic Service (EBS)* für Mörsergeschosse und der *LFK-Dienst* für Lenkflugkörper genannt (vgl. *direct fire, indirect fire* und *smart weapon* in Abschnitt 2.5.1).

Beispiel eines Schadensservices Ein Beispiel für einen Effekt- oder Schadensservice ist der *Weapon Effect Service (WES)*, der von der Firma IABG entwickelt wurde und vertrieben wird. „Der WES wird verwendet, um einheitliche Waffeneffekte für alle beteiligten Föderaten einer Simulation zu berechnen. Ausgelöst durch eine *Munition Detonation Interaction* des RPR-FOM in der HLA werden die Schadenseffekte bewertet. Die Ergebnisse der Bewertung werden über die HLA per *Weapon Effect Interaction* (Erweiterung des

RPR-FOM) publiziert. Die beteiligten Simulationssysteme könnten die interne Auswertung von Schadenseffekten deaktivieren und die Schadensauswirkungen aus der übermittelten *Weapon Effect Interaction* ableiten“ (frei übersetzt aus [59, 91]).

2.7.4 Ergebnisse des Projekts VIntEL

Grundsätzlich ist mit den Ergebnissen von VIntEL eine Steigerung der Interoperabilität und eine Verbesserung des Fair-Fight in der verteilten, virtuellen Simulation verbunden [45, 51, 72, 81, 82].

Lüthi [52] erkannte im Jahr 2011, dass mit der Architektur von Effektservices in VIntEL zwar nach LCIM (vgl. Abschnitt 2.4.3) die Ebene der semantischen Interoperabilität erreicht wird, die Bedingungen für Interoperabilität auf pragmatischer Ebene jedoch nicht erfüllt sind. Dies gilt insbesondere für die Gebäudeschäden, denn „[...] welche Konsequenz sich beim empfangenden Förderaten aus einer berechneten Trefferwirkung ergibt, ist Frage der pragmatischen Ebene.“ [52]

Konkret wird in [52] aufgezeigt, dass die prozentuale Schadenklassifikation zu einem gleichen Verständnis des Informationsaustausches führt und hinreichend für die semantische Interoperabilität sind. Das Level der pragmatischen Interoperabilität jedoch nicht erfüllt ist, da die explizite Darstellung des Schadens weiterhin dem einzelnen Simulationssystem obliegt. Da der Schaden in jedem Simulationssystem anders dargestellt werden kann, widerspricht dies der Eindeutigkeit von Methoden und Funktionen und somit der pragmatischen Interoperabilität.

Obwohl, zeitlich gesehen nach den Bewertungen von Lüthi [52] im Jahr 2011, die Entwicklung der in Abbildung 2.15 dargestellten zentralen ISO OGC konformen Datenbasis (vgl. [45, 47, 81, 82]) durchgeführt wurde, bleibt die dargestellte Handhabung der Gebäudeschäden steigerungsfähig: Zwar kann der Gebäudezustand einheitlich als prozentualer Schaden modelliert werden (vgl. Abbildung 2.14), das ursächliche Ereignis, das zur Beschädigung des 3D-Gebäudemodells geführt hat, bleibt jedoch unberücksichtigt. Eine vorgerechnete prozentuale Schadenklassifikation kann zwar den allgemeinen Zustand eines 3D-Gebäudemodells darstellen, die echte Wechselwirkung, die während der Simulation aufgetreten ist, wird jedoch nicht wiedergegeben.

Kapitel 3

Anforderungen und alternative Lösungswege

In diesem Kapitel wird der Begriff des Gebäudeschadens definiert und die Anforderungen zu Handhabung von Gebäudeschäden näher erläutert. Anschließend werden diese Anforderungen mit alternativen Lösungswegen verglichen.

3.1 Definition von beschädigten Gebäuden

Ein beschädigtes Gebäude, im Weiteren auch Gebäudeschaden, wird in der verteilten, virtuellen Simulation als optisches Feedback eines Waffeneinsatzes für den Übenenden benötigt. Durch den militärischen Kontext und unter der Rahmenbedingung, dass in dieser Arbeit nur Simulationssysteme bemannter Waffensysteme fokussiert werden, werden beschädigte Gebäude wie folgt definiert:

1. **Ein beschädigtes Gebäude ist ein geometrisch verändertes 3D-Gebäudemodell.** Auswirkungen auf die etwaige Semantik des 3D-Gebäudemodells (Beispielsweise aus *Dach* wird *Schutthaufen*) werden nicht thematisiert, da es für das benötigte optische Feedback nicht von primärer Bedeutung ist. Mit dieser Definition ist verbunden, dass das beschädigte Gebäude ein stabiler Zustand ist. Das beschädigte Gebäude ist kein animationsartiger Veränderungsprozess von 3D-Gebäudemodellen, sondern der endgültige Zustand nach einer geometrischen Veränderung.
2. **Es werden nur geometrische Veränderungen betrachtet, die unmittelbar auf den direkten, indirekten oder präzisionsgelenkten Einsatz von Waffensystemen (vgl. Abschnitt 2.5.1) zurückzuführen sind.** Es werden nur die Waffensysteme berücksichtigt, die beispielsweise durch Schusswaffen („Schusswaffen: Geschütz, Granatwaffe und Handfeuerwaffe“ [95]) oder Raketenwaffen geometrische Veränderungen verursachen können. Geometrische Veränderungen eines Gebäudes, die beispielsweise durch Naturkatastrophen, Alterung oder Feuer verursacht werden,

werden nicht betrachtet, da diese im Kontext von virtuellen Simulationen bemannter Waffensysteme nicht im Fokus stehen.

3.2 Anforderungen zur Handhabung von Gebäudeschäden

In diesem Abschnitt werden die in der Einleitung genannten Ziele (vgl. Abschnitt 1.2) zu konkreten Anforderungen an die Handhabung von Gebäudeschäden erweitert.

3.2.1 Authentizität

Ein Gebäudeschaden muss nach einem Waffeneinsatz in einem Simulationssystem so dargestellt werden, dass sich der Übende weiterhin oder trotzdem mit dem Geschehen in der virtuellen Welt der Simulation identifizieren kann. Der Gebäudeschaden muss authentisch sein.

Jeder Übende entwickelt vor dem Abschuss der Waffe eine Vorstellung der Waffenwirkung. Diese Vorstellung basiert auf den ihm zur Verfügung stehenden Parametern. Diese Parameter betreffen das Gebäude (beispielsweise Form und Größe des Gebäudes) und den Waffeneinsatz (beispielsweise Waffentyp, Entfernung zum Ziel, Munitionsart) und weitere äußere Randbedingungen. Ein Gebäudeschaden wird vom Übenden genau dann für authentisch gehalten, wenn der Gebäudeschaden mit seiner Vorstellung von der Waffenwirkung übereinstimmt. Dies bedeutet im Umkehrschluss, dass genau diese zur Verfügung stehenden Parameter des Gebäudes und des Waffeneinsatzes für die Erzeugung des Gebäudeschadens verwendet werden müssen. Ein authentischer Gebäudeschaden basiert auf den Eigenschaften des Gebäudes und auf den Eigenschaften des Waffeneinsatzes.

Die weiteren äußeren Randbedingungen, die die Vorstellung des Übenden beeinflussen könnten, wie beispielsweise die aktuelle Witterung oder die Geländetopographie, stehen in der vorliegenden Arbeit nicht im Fokus.

Subjektive oder persönliche Einflüsse, wie die Erfahrung oder die Persönlichkeit, sind unmittelbar vom Übenden abhängig und können nur berücksichtigt werden, indem eine bestimmte Vorbildung der Übenden angenommen wird. Es sei angenommen, dass in der virtuellen Simulation nur Personen mit militärischer Prägung (beispielsweise Ausbildung und Erfahrung in den Themengebieten Waffensysteme, Sprengstoff und Ballistik) agieren, die eine realistische Vorstellung von Waffenwirkungen mitbringen.

3.2.2 Konsistenz

Die Berechnung des Gebäudeschadens muss konsistent sein.

Konsistenz bedeutet nicht, dass ein Gebäudeschaden in jedem Simulationssystem einer verteilten Simulation zwingend gleichartig visualisiert wird. Konsistenz bedeutet, dass der **Berechnungsprozess** nicht zu inkonsistent beschädigten Gebäuden führt.

Dies wird über die Verwendung des LoD-Konzepts erläutert (vgl. Abschnitt 2.6.3). Durch das LoD Konzept ist es üblich, dass für jedes 3D-Gebäudemodell mehrere Instanzen unter-

schiedlicher grafischer Detaillierung bestehen, welche sichtdistanzabhängig ein- oder ausgeblendet werden. Ein Gebäudeschaden muss jedoch nicht zwingend in allen LoD eines 3D-Gebäudemodells repräsentiert sein. Insbesondere kleine Gebäudeschäden werden nur in detailreichen LoD benötigt.

Wird nun beispielsweise der Schornstein eines 3D-Gebäudemodells beschädigt und ist der unbeschädigte Schornstein nur im detailreichsten LoD dargestellt, so darf auch der beschädigte Schornstein nur im detailreichsten LoD visualisiert werden. Würde der Gebäudeschaden in allen LoD dargestellt, führt dies indirekt dazu, dass durch den Gebäudeschaden in den nicht detailreichsten LoD ein beschädigter Schornstein entstehen würde, obwohl vorher kein Schornstein vorhanden war. Durch den Berechnungsprozess würden Objekte (Schornsteine) entstehen, die vorher nicht vorhanden waren. Die Berechnung wäre nicht konsistent.

Die Anforderung der Konsistenz kann auch am Beispiel unterschiedlicher Simulationssysteme erläutert werden. Wenn beispielsweise die Dachgaube eines Gebäudedaches in einem Fahrsimulator leicht beschädigt wird, in einem verbundenen Flugsimulator jedoch grundsätzlich keine Dachgauben dargestellt werden, und nur grafisch vereinfachte Dachformen repräsentiert werden, darf die Beschädigung nicht dazu führen, dass im Flugsimulator eine Gaube entsteht.

Die geometrischen Repräsentationsunterschiede zwischen LoD in **einem** Simulationssystem müssen durch die Berechnung des Gebäudeschadens ebenso erhalten bleiben, wie die geometrischen Repräsentationsunterschiede zwischen **unterschiedlichen** Simulationssystemen (Beispiel Dachgaube).

3.2.3 Reaktionszeit

Der Gebäudeschaden muss in Echtzeit berechnet und visualisiert werden.

In der Realität wird die Sichtbarkeit eines Gebäudeschadens nach einer Waffenwirkung durch Rauch und Staubeentwicklung gestört. Es dauert einige Sekunden bis ein Gebäudeschaden sichtbar wird. In der Simulation steht nur dieser Zeitraum von wenigen Sekunden zur Verfügung, um den Gebäudeschaden am 3D-Gebäudemodell zu berechnen und darzustellen. Da der Übende eine Reaktion des Waffeneinsatzes erwartet, führen längere Reaktionszeiten zu einem Verlust der Authentizität der Simulation. Welche Reaktionszeit gefordert ist, hängt von der übenden Person ab und kann letztlich nicht objektiv beurteilt werden.

Es ist aber möglich einen groben Richtwert zu ermitteln, wie lange in etwa die Sichtbarkeit des Gebäudeschadens nach einer Waffenwirkung durch Rauch- oder Staubeentwicklung gestört wird. Aus verschiedenen Videos [98] realer Waffeneinsätze wird näherungsweise abgeleitet, dass mindestens 5 Sekunden nach der Waffenwirkung keine Gebäudeschäden sichtbar sind. Ausschnitte der Videos, die jeweils den Zeitpunkt der Detonation und die Rauch und Staubeentwicklung 5 Sekunden nach der Detonation zeigen, sind in Abbildung 3.1 dargestellt. In der Abbildung ist erkennbar, dass die Sicht 5 Sekunden nach der Deto-

nation noch verdeckt ist.

Auf Basis dieser Videos wird für die vorliegende Arbeit eine Reaktionszeit von etwa 5 Sekunden als Anforderung der Reaktionszeit angenommen. Innerhalb von 5 Sekunden muss ein Gebäudeschaden nach einer Waffenwirkung berechnet und dargestellt werden.



Abbildung 3.1: Eingeschränkte Sichtbarkeit nach Detonationen [98]. Obere Reihe: Zeitpunkt der Detonation. Untere Reihe: Etwa 5 Sekunden nach der Detonation.

3.2.4 Transparenz

Die Berechnungsschritte zur Erzeugung des Gebäudeschadens müssen nachvollziehbar sein.

Die Anforderung der Transparenz geht über die Beschreibung von Arbeitsschritten hinaus. Mit dieser Anforderung ist der allgemeine Grundsatz der Bundeswehr verbunden, die Kernkompetenzen über Simulationsprozesse zu wahren. Mit Simulationen können allgemeine Rückschlüsse auf die unmittelbare Leistungsfähigkeit oder das Verhalten von Einheiten in spezifischen Situationen gezogen werden. Diese Fähigkeit einer Bewertung von Simulationsergebnissen muss innerhalb der Bundeswehr verbleiben. Dies wird als die Anforderung der „amtsseitigen und nationalen, unabhängigen Urteilsfähigkeit“ [86] bezeichnet.

Ein neuer Ansatz zur Handhabung von Gebäudeschäden muss dieser Anforderung nachkommen. Zu jedem Gebäudeschaden müssen die verwendeten Parameter und Algorithmen bekannt sein. Die Dokumentation und Offenlegung des Quellcodes ist gefordert.

Darüber hinaus muss die Handhabung von Gebäudeschäden erweiterbar und veränderbar sein. Eine technische Umsetzung dieser Anforderung ist die Bereitstellung einer API (Programmierschnittstelle - *Application Programming Interface*).

3.2.5 Integrierbarkeit

Die Handhabung von Gebäudeschäden muss als Modul in unterschiedliche verteilte Simulationen integrierbar sein.

Je nach Übungsszenario bestehen verteilte Simulationen aus unterschiedlichen Simulationssystemen. Die teilnehmenden Simulationssysteme von Joint Exercises variieren. Die neue Handhabung von Gebäudeschäden darf nicht auf einzelne Joint Exercises beschränkt oder von konkreten Simulationssystemen abhängig sein.

Diese Anforderung entspricht dem Konzept des M&S as a Service (vgl. Abschnitt 2.4.6). Die Handhabung von Gebäudeschäden muss als Service in verschiedene verteilte Simulationen eingebunden werden können.

3.2.6 Bestehende Standards

Etablierte Vorgehensweisen und Standards der verteilten Simulation sind zu berücksichtigen.

Im vorherigen Kapitel wurden unterschiedliche Normen und Standards zur Modellierung und Vorhaltung von 3D-Gebäudemodellen in der virtuellen Simulation vorgestellt (vgl. Abschnitt 2.6). Eine neue Handhabung von Gebäudeschäden muss für 3D-Gebäudemodelle unterschiedlicher Normen und Standards verwendbar sein.

Darüber hinaus umfasst die Forderung der Nutzung bestehender Standards auch die Berücksichtigung etablierter Techniken der verteilten Simulation. Der Datenaustausch der verteilten Simulation per DIS oder HLA ist eine solche etablierte Technik (vgl. Abschnitt 2.4). Eine neue Handhabung von Gebäudeschäden darf keine gänzlich neue Architektur für verteilte Simulationen propagieren.

3.3 Bestehende Ansätze von Gebäudeschäden

Die Veränderungen von virtuellen Gebäudedarstellungen sind kein Alleinstellungsmerkmal der virtuellen Simulation. Es existieren alternative Ansätze zur Handhabung von Gebäudeschäden. Im Folgenden werden die Anforderungen der verteilten, virtuellen Simulation mit den Bereichen der 3D-Geoinformationssysteme, der realphysikalischen Simulationen und den Möglichkeiten aktueller Videospiele verglichen.

3.3.1 3D-Geoinformationssysteme

Geoinformationssysteme mit der Fähigkeit der Datenhaltung und Visualisierung von dreidimensionalen Geometriedaten werden als *3D-GIS* bezeichnet. Die häufigste Anwendung von 3D-GIS ist die Datenhaltung, -aufbereitung und -visualisierung von virtuellen 3D-Stadtmodellen. Die darstellbaren Details von Stadtmodellen in 3D-GIS gehen heutzutage weit über die ersten als Klötzchenmodelle bekannten Visualisierungen hinaus und sind kaum von echten Gebäuden zu unterscheiden (vgl. Abbildung 3.2). [42, 43]

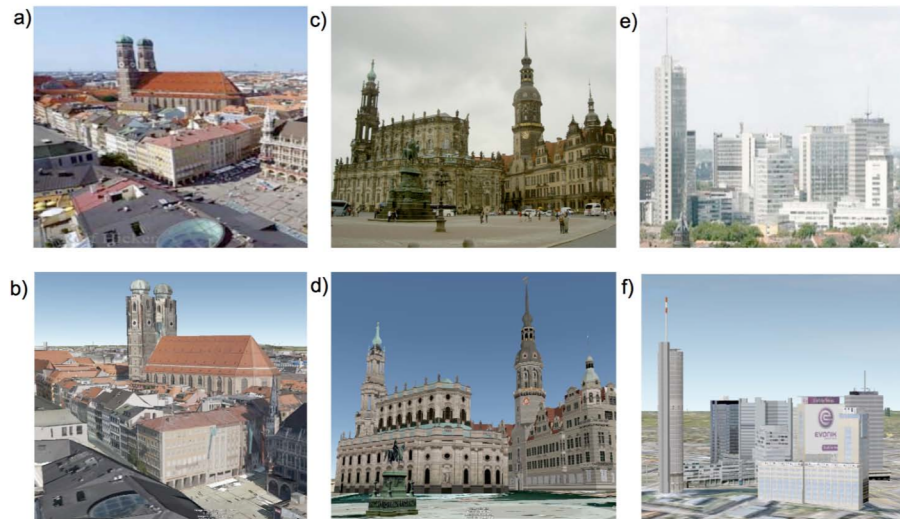


Abbildung 3.2: Gegenüberstellung von Fotoaufnahmen (oben) und virtuellen Stadtmodellen (unten) aus [43].

Neben den prominenten Vertretern der virtuellen Globen, wie Google Earth [29] oder Nokia Maps [60], werden 3D-GIS zur Darstellung und Datenhaltung von kleinräumigen oder urbanen Gebieten verwendet. Diese lokalen 3D-Stadtmodelle¹ stehen zumeist auf kommunaler Ebene zur Verfügung und werden unter der Schirmherrschaft der zuständigen Katasterämter erstellt und fortgeführt. Auch wenn die Leistung dieser Systeme im Hinblick auf Realismus und Mengengerüst den darstellbaren Objekte den virtuellen Globen nachsteht, ist deren funktionaler Umfang ungleich höher. Insbesondere semantikbasierte Auswertungsmethoden kennzeichnen diesen Mehrwert. Die Basis für die Auswertemethoden bilden zumeist generische Datenstrukturen, offene Standards und die Austauschbarkeit durch sogenannte Geodateninfrastrukturen. Die virtuelle Einblendung von Planungsvorhaben oder Solarpotenzialen von Dächern sind nur zwei Beispiele für die Verwendung von kommunalen 3D-GIS. Eine Übersicht zu verschiedenen Anwendungen von kommunalen 3D-Stadtmodellen ist in der Informationsbroschüre der gemeinsamen Kommission 3D-Stadtmodelle der DGPF und DGfK [1] dargestellt.

Die lokalen 3D-GIS dienen der öffentlichen Daseinsvorsorge und der Aufgabenerfüllung von planungs- und baubehördlichen Tätigkeiten. Die Amtlichkeit, also die Qualität und Aktualität, ist im Hinblick auf Verlässlichkeit und Genauigkeit des 3D-Stadtmodells höher zu bewerten als die authentische Visualisierung. Ein 3D-Gebäudemodell muss nicht vom Bürger für authentisch gehalten werden, sondern soll in Form und Größe den aktuellen, ortsüblichen Gegebenheiten entsprechen.

Die unmittelbare Handhabung von Gebäudeschäden spielt bei 3D-GIS insofern eine Rolle, als dass der (Teil-) Abriss, der Neubau oder die Veränderung von Gebäuden auch zeitnah in den Datenbestand übernommen werden muss. Der Fortführungsprozess der 3D-Daten muss

¹Beispiele für kommunale 3D-Stadtmodelle die in einem 3D-GIS vorgehalten werden: **Stadt Dortmund:** <https://www.dortmund.de> - letzter Aufruf 07.05.2018 **Stadt Stuttgart:** <https://3d.stuttgart.de> - letzter Aufruf 07.05.2018

gesichert sein und wird im 3D-GIS durchgeführt. Der betrachtete Zeitraum der Fortführung entspricht den realen Bau- oder Abrissprozessen und kann mit Wochen und Monaten angegeben werden. Es wird nicht jede tägliche Veränderung eines Bauprozesses in den Datenbestand überführt. Erst nach Abschluss einer Veränderungsmaßnahme erfolgt die Einmessung (Digitalisierung der Veränderung) und die Übernahme in den Datenbestand. Als Austauschformat für Datenbestände kommunaler 3D-GIS kann dem CityGML-Standard (vgl. Abschnitt 2.6.3), zumindest im deutschen Raum, ein Alleinstellungsmerkmal zugesprochen werden.²

Die grundsätzliche Durchdringung von OGC Standards in den Fortführungsprozessen, die ISO und OGC Konformität in der Datenhaltung und die Verwendung von Servicearchitekturen sind bei den Datenhaltungen von amtlichen Geodaten verpflichtend vorgeschrieben (vgl. *Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)* [24]). Die Standardisierung ist in (3D-) Geoinformationssystemen Status quo.

3.3.2 Realphysikalische Simulationen

Als realphysikalisch werden diejenigen Simulationen verstanden, deren Ziel die Verhaltensprädiktion von Bauteilen unter dem Einfluss physikalischer Kräfte ist. Die virtuelle Zerlegung des Bauteils in infinitesimale Bestandteile ist eine Möglichkeit, diese Prädiktion zu berechnen und unter dem Namen der finiten Elemente bekannt. „Die *Finite Elemente Methode (FEM)* [...] ist eine weit verbreitete numerische Lösungsmethode im Bereich wissenschaftlich technischer Aufgabenstellungen. Mit der FEM können physikalische Vorgänge (beispielsweise Kraftwirkungen auf deformierbare Festkörper) simuliert werden, deren Verlauf sich nicht oder nur sehr aufwendig mit anderen Mitteln bestimmen lässt“ [95].

Die FEM wird zur Simulation des realistischen Verhaltens von Bauteilen unter physikalischen Einflüssen wie Massen oder Beschleunigungen verwendet. Die Anwendungsszenarien dieser Simulationen sind vielfältig. Von der Statik einzelner Bauelemente über Tragfähigkeitsanalysen von Brückenbauwerken bis hin zur simulationsbasierten Prädiktion von Hohlräumen nach Naturkatastrophen³ wird die FEM verwendet. Das Ziel der Simulationen ist stets die korrekte Vorhersage der Auswirkungen der physikalischen Einflüsse auf das Bauteil oder das Objekt.

Im Hinblick auf die Anforderungen der Gebäudeschäden steht die authentische Visualisierung bei der FEM nicht im Fokus. Nicht die visuelle Darstellung eines zerberstenden Stahlträgers wird benötigt, sondern die numerischen Werte bei der die Tragfähigkeit des Stahlträgers signifikant beeinträchtigt wird.

Eine weitere Eigenschaft der realphysikalischen Simulation ist die Abgeschlossenheit und Proprietät. Bautechnische oder statische Fragestellungen, wie die Tragfähigkeit eines Stahl-

²CityGML ist beispielsweise Referenzstandard für 3D-Stadtmodelle der AdV [2].

³Beispielsweise wird im EU Projekt INACHUS, der Einsturz von Gebäude nach Naturkatastrophen simuliert, um Hohlräume mit möglichen Überlebenden zu finden <http://www.inachus.eu/> -letzter Zugriff 07.05.2018.

Veröffentlichung	Prozessorspezifikation	Berechnungszeit
Mattern et al. [54] 2007	8 Kerne 1,5 GHz	18 Stunden
Griffin [30] 2008	2 Kerne 2,4 GHz	55 Stunden
Lee und E-Tawil [48] 2008	1 Kern 1,73 GHz	8 Minuten (CPU Zeit)
Sikiwat et al. [74] 2009	1 Kern 1,6 GHz	7 Stunden
Ali et al. [3] 2011	2 Kerne 2 GHz	15 Stunden

Tabelle 3.1: Vergleich der Berechnungszeiten von realphysikalischen Schadensvisualisierungen aus [3] (frei übersetzt).

trägers, sind sehr speziell und vom individuellen Bauprojekt abhängig. Bauwerke, bei denen sich die Rahmenbedingungen so weit ähneln, dass Interoperabilität oder Austauschbarkeit von Simulationsmodellen der FEM möglich wären, müssten quasi identisch sein. Da es sich bei Bauwerken jedoch zumeist um individuelle Objekte handelt, sind identische Rahmenbedingungen unwahrscheinlich. Realphysikalische Simulationen werden für Einzelfälle und spezifische Projekte angefertigt und berechnet. Unabhängige Standardisierungsbestrebungen von Datenmodellen oder Austauschstandards der FEM sind nicht bekannt.

Die FEM kann zur Erzeugung von Gebäudeschäden eingesetzt werden. In Ali et al. [3] werden Ansätze zur Berechnung von Gebäudeschäden auf FEM Basis verglichen und vorgestellt. Die Ergebnisse dieses Vergleichs sind in Tabelle 3.1 abgebildet. Darin ist auffällig, dass die Berechnungs- und Visualisierungszeiten mindestens im Minuten- wenn nicht sogar im Stundenbereich liegen. Diese Eigenschaften der Gebäudeschäden von FEM basierten Berechnungen sind mit den Anforderungen der verteilten, virtuellen Simulation (Reaktionszeit 5 Sekunden, vgl. Abschnitt 3.2.3) nicht vereinbar.



Abbildung 3.3: Realphysikalische Simulation eines Gebäudeschadens erstellt mit der Designsoftware *Blender* aus [98] (URL: <https://www.youtube.com/channel/UCdiTXx3e55HrQepgWj2Tt3w> - letzter Zugriff: 07.05.2018).

Neben der FEM gibt es weitere Möglichkeiten zur Berechnung realphysikalische Gebäudeschäden. Eine bekannte Möglichkeit ist, Gebäude durch eine Vielzahl identischer starrer Körper darzustellen (Analogie zum Brett- und Geschicklichkeitsspiel *Jenga*⁴). Im Unter-

⁴„Jenga ist ein Geschicklichkeitsspiel. Es besteht aus 60 [...] gleichen hölzernen Bauteilen in Quaderform [...], die zu Beginn des Spiels zu einem Turm gestapelt werden, indem immer drei Bausteine nebeneinander zu liegen kommen.“ [95]

schied zur FEM werden die starren Körper ohne gegenseitige Verbindungen modelliert und durch Gewicht und Schwerkraft gehalten. Wenn die gestapelten Körper durch weitere Kräfte oder Beschleunigungen beeinträchtigt werden, führt dies zum Einsturz des Stapels und zu einem Gebäudeschaden. Dies ist in Abbildung 3.3 dargestellt. Das 3D-Gebäudemodell in Abbildung 3.3 besteht aus mehr als 2000 einzelnen Quadern (starren Körpern), mit jeweils eigenem physikalischen Verhalten. Die Rechenzeit der Simulation beträgt etwa 7 Stunden. Zwar können mit dieser Methode auch authentische Gebäudeschäden erzeugt werden, allerdings liegt auch bei dieser Methode die Rechenzeit im Stundenbereich. Die Herstellung der stapelbasierten 3D-Gebäudemodelle aus Körpern, wie Quadern oder Würfeln, ist zusätzlicher Aufwand.

3.3.3 Videospiele

In Videospiele der sogenannten Egoshooter-Kategorie bewegt sich ein Spieler virtuell durch eine dreidimensionale Welt. Neben der typischen Waffenwirkung gegen andere Spieler oder computergesteuerte Gegner, ermöglichen moderne Egoshooter auch die Bewirkung der Umwelt. *Farcry 3* oder *Battlefield 4* sind Beispiele für kommerzielle Videospiele, in denen auch die Veränderung von Gebäuden möglich ist. Innerhalb dieser Spiele können Gebäudeschäden erzeugt werden.

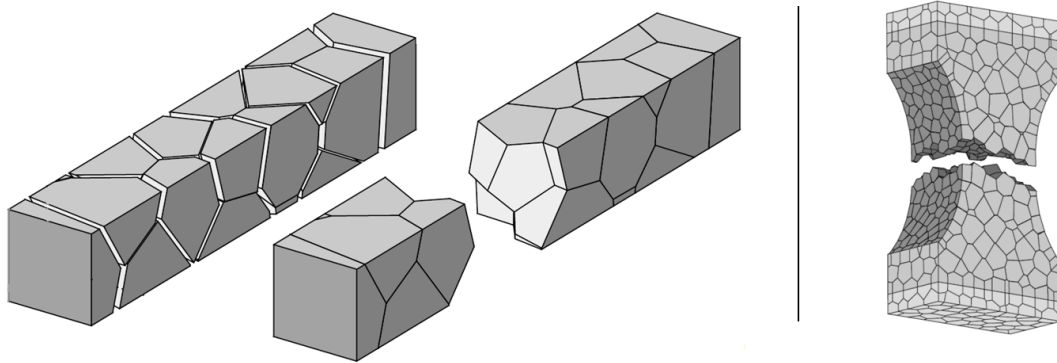


Abbildung 3.4: Beispiele des Voronoi Fracturing. Links: Zwei Darstellungen eines grob fragmentierten Quaders. Rechts: Feinere Fragmentierung eines komplexeren 3D-Objekts. Abbildungen aus [83] (verändert).

Methodisch wird für diese Erzeugung der Gebäudeschäden und insbesondere die Zerlegung von Geometrie das sogenannte *Voronoi Fracturing* verwendet. Das Verfahren nach Sukumar und Bolander [83] beschreibt die Zerlegung von 3D-Geometrien in eine Vielzahl von zusammenhängenden Volumenkörpern. Diese Zerlegung ist in Abbildung 3.4 dargestellt. Für alle neu entstehenden Volumenkörper werden Beschleunigungsanimationen berechnet und visualisiert, die vom Explosionsort radial nach außen gerichtet sind. Es entsteht der Eindruck, dass das Gebäude in Trümmerteile zerlegt wird und die Trümmer radial vom

Einschlagspunkt versprengt werden. Diese Zerlegung mit anschließender radialer Beschleunigung führt zu einem fragmentierten 3D-Gebäudemodell. Eine geringe Anzahl von Fragmenten erlaubt beim Voronoi Fracturing eine nicht wahrnehmbare Reaktionszeit, so dass die Fragmentierung und Beschleunigung zur Laufzeit des Videospieles berechnet und visualisiert werden kann. Die Zeitreihe eines auf Voronoi Fracturing basierten Gebäudeschadens zeigt Abbildung 3.5.



Abbildung 3.5: Zeitreihe der Erzeugung eines Gebäudeschadens per Voronoi Fracturing aus [98] (URL: <http://www.youtube.com/watch?v=pGn3kNCPpHk> letzter Zugriff: 12.05.2018).

Eine für die militärische Simulation relevante Mischung aus Simulation und Videospiele ist das Produkt *Virtual Battlespace (VBS)* der Firma *Bohemia Interactive Simulations* [92]. VBS ist eine flexible Lösung für militärische Trainingsanwendungen und bietet die Möglichkeit, virtuelle 3D-Darstellungen für unterschiedlichste Simulationen bereitstellen zu können [92]. Die Versionen VBS 2 und VBS 3 sind auch bei der Bundeswehr im Einsatz. „Bei dem Trainingsprogramm VBS handelt es sich um ein *Serious Game*. Unter dem Begriff *Serious Games* versteht man – ursprünglich für den zivilen Markt erstellte – Computerspiele, deren primäres Ziel nicht der Unterhaltung dient, sondern deren Ziel auf den militärischen Bedarf und, im Rahmen einer militärischen Nutzung, die Vermittlung von Wissen, Fertigkeiten und Verfahren ist. Die *Serious Games* sind in Einzel- oder Teamausbildung zu pädagogischen Zwecken in der Ausbildung verwendbar. Sie sind insbesondere für eine handlungsorientierte Ausbildung geeignet und vermitteln Kompetenzen in den Bereichen Führung, Entscheidung und Zusammenarbeit im Team“ [9] (orthographisch verändert). Beispiele für ursprüngliche Computerspiele oder 3D-Engines, die vollständig oder teilweise in der militärischen Nutzung virtueller Simulationen verwendet werden, sind neben VBS die *CryEngine* der Firma *Crytek*, das Computerspiel *Arma: Combat Operations* von *Bohemia Interactive* oder die *Havok Vision Game Engine* des Herstellers *Trinigy*. In VBS 3 ist auch die Erzeugung von Gebäudeschäden möglich. Neben der herkömmlichen Methode der *Level of States* können individuelle Gebäudeschäden zur Laufzeit berechnet werden. Wie in Abbildung 3.6 erkennbar ist, setzt VBS hierbei ebenfalls auf die Methode des Voronoi Fracturing. Nach Aussage von *Eurosimtec*, dem deutschen Vertriebspartner für VBS, handelt es sich jedoch nicht um eine allgemeine und generische Lösung für Gebäudeschäden. Nur spezielle und aufwendig aufbereitete Gebäude von VBS erlauben die Erzeugung von Gebäudeschäden⁵. Die exakten Parameter und Berechnungsschritte, die zum Gebäudeschaden führen, bleiben der internen Logik von VBS vorbehalten und widersprechen somit der Anforderung der Transparenz (vgl. Abschnitt 3.2.4).

⁵Quelle dieser Aussage ist ein telefonisches Interview mit Eurosimtec am 29.09.2016

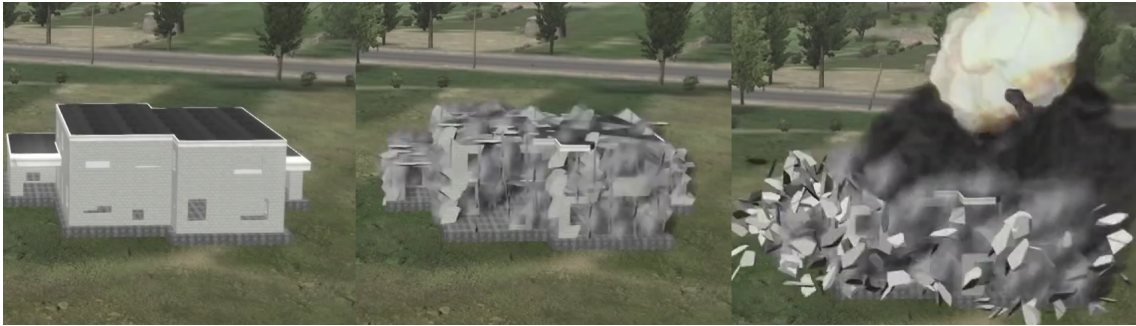


Abbildung 3.6: Zeitreihe der Erzeugung eines Gebäudeschadens von einem speziell aufbereiteten 3D-Gebäudemodell in VBS 3 (Abbildungen aus [92]).

Obwohl durch die Methode des Voronoi Fracturing optisch beeindruckende Gebäudeschäden berechnet und dargestellt werden können, sind diese nicht zwingend authentisch. Die Anzahl der Fragmente muss gering gehalten werden, damit das anschließende Beschleunigen der Trümmerfragmente physikalisch korrekt zur Laufzeit berechnet werden kann. Je mehr Fragmente erzeugt werden, desto aufwendiger ist die anschließende Berechnung der Bewegungswege der Trümmerteile. Diese These wird durch den Benchmark von Mondesire et al. aus dem Jahr 2016 bestätigt [56].⁶

Die zuvor dargestellten Gebäudeschäden per Voronoi Fracturing sind Bestandteil kommerzieller Videospiele oder sogenannter Serious Games. Servicebasierte oder standardisierte Handhabungen von Gebäudeschäden stehen jeweils nicht im Fokus. Da ein Videospiele stets als abgeschlossenes Produkt entwickelt und vertrieben wird, ist die Kopplungsfähigkeit zwischen unterschiedlichen Videospiele (Anforderung der Integrierbarkeit vgl. Abschnitt 3.2.5) weder funktional gefordert noch kommerziell attraktiv.

3.3.4 Zusammenfassung

In Tabelle 3.2 werden die Anforderungen zur Handhabung von Gebäudeschäden dieser Arbeit mit den zuvor vorgestellten Möglichkeiten der 3D-GIS, realphysikalischen Simulationen und den Videospiele verglichen. Entsprechend der Anforderungen finden sich in der Tabelle die Kriterien *Authentizität*, *Konsistenz*, *Reaktionszeit*, *Transparenz*, *Integrierbarkeit* und *Bestehende Standards*.

Die Möglichkeiten der Darstellung von Gebäudeschäden in 3D-GIS sind bezüglich der Reaktionszeit (Tage bis Wochen) nicht für die Anforderungen der virtuellen Simulation geeignet. Zwar erlauben 3D-GIS authentische Darstellungen eines 3D-Gebäudebestandes, aber deren

⁶In den Tests führender physikalischer Berechnungssoftware, den sogenannten *Physikengines*, in [56] werden die Performanz und Leistungsfähigkeit der offenen *Open Dynamics* und *Bullet Engine* sowie der kommerziellen *PhysX Engine* von NVidia verglichen. Auch wenn die Leistungen der Engines unterschiedlich sind, vereint sie das generelle Limit der parallel handhabbaren Primitiven (vergleichbar zu den Fragmenten eines Gebäudeschadens). In einem Testszenario wurde untersucht, ab welcher Anzahl geometrischer Primitive die Bildwiederholrate (vgl. Abschnitt 2.2.3) unter 10 FPS bleibt, also die Berechnungsdauer länger als 100 Millisekunden dauert. Dabei erwiesen sich bei der Open Dynamics Engine bereits 200 Objekte als kritisch, wohingegen die PhysX Engine etwa 3300 Objekte berechnen konnte. Kernaussage ist, dass die Leistungsfähigkeit aller getesteten Engines auch bei moderner Hardware begrenzt bleibt.

kurzzeitige Veränderung ist nicht möglich. Von Interesse sind hingegen Standardisierung der Datenhaltung sowie die Fortführung von 3D-Gebäudemodellen auf Basis von Normen und Standards und die damit verbundene Transparenz sowie Verwendung bestehender Standards.

Realphysikalische Simulationen zeichnen sich durch die Exaktheit der physikalischen Phänomene und der damit verbundenen Qualität aus. Problematisch sind hingegen die Anforderungen bezüglich Standardisierung, Integrierbarkeit, Authentizität und Reaktionszeit. Standardisierung und Integrierbarkeit sind auf Grund des konkreten und projektbezogenen Charakters nicht relevant. Ebenso sind die langen Reaktions-, also Berechnungszeiten, für die physikalische Korrektheit zwar nötig, für die Anforderungen der virtuellen Simulation aber ungeeignet. Die Authentizität von Gebäudeschäden ist ebenfalls kein Ziel von realphysikalischen Simulationen.

Gebäudeschäden in 3D-Videospielen liefern beeindruckende Darstellungen eines Ereignisses mit einer minimalen Reaktionszeit. Vergleichbar zu realphysikalischen Simulationen, finden Integrierbarkeit und Standardisierung jedoch keine Verwendung.

Zusammengefasst existiert in keiner der drei betrachteten Ansätze eine Lösung zur Handhabung von Gebäudeschäden, die den Anforderungen der verteilten, virtuellen Simulation vollumfänglich genügen. In jedem Bereich können sowohl Vor- als auch Nachteile identifiziert werden. Eine ganzheitlich adaptierbare Strategie für die verteilte, virtuelle Simulation existiert nicht.

Anforderungen	3D-GIS	realphysikalische Simulation	3D-Videospiel
Authentizität	-	-	+
Konsistenz	-	+	-
Reaktionszeit	-	-	+
Transparenz	+	+	-
Integrierbarkeit	-	-	-
Bestehende Standards	+	-	-

Tabelle 3.2: Vergleich der Anforderungen der Handhabung von Gebäudeschäden der verteilten, virtuellen Simulation mit den Möglichkeiten von 3D-Geoinformationssystemen, realphysikalischen Simulationen und 3D-Videospielen.

Kapitel 4

Ein neues Verfahren zur Handhabung von Gebäudeschäden

In diesem Kapitel wird ein neues Verfahren zur Handhabung von Schadensereignissen zur Erzeugung von beschädigten Gebäuden in der verteilten, virtuellen Simulation vorgestellt. Zur Herleitung des Verfahrens wird die Handhabung von Schadensereignissen als Prozesskette beschrieben.

4.1 Prozessorientierte Darstellung zur Erzeugung von beschädigten Gebäuden

Die Erzeugung von beschädigten Gebäuden wird als Prozess approximiert. In Abbildung 4.1 ist der Prozess dargestellt. Der Prozess basiert auf der Definition für beschädigte Gebäude aus Abschnitt 3.1. Die verschiedenen Bestandteile und Bezeichnungen in der prozessartigen Darstellung (vgl. Abbildung 4.1) werden in den folgenden Abschnitten beschrieben und diskutiert.

Unbeschädigtes Gebäude Das *unbeschädigte Gebäude* ist das 3D-Gebäudemodell vor dem Einfluss des Schadensereignisses. Das unbeschädigte Gebäude wird auch als Ausgangszustand bezeichnet.

Eine Gemeinsamkeit aller unter Abschnitt 2.6.6 dargestellten Normen und Standards für 3D-Gebäudemodelle in der virtuellen Simulation ist die Verwendung der Boundary Representation (vgl. Abschnitt 2.6.1). Sowohl CityGML, SEDRIS als auch Open Flight verwenden BRep zur Definition von Geometrie. Es wird daher angenommen, dass die Geometrie des unbeschädigten Gebäudes der Boundary Representation entspricht. Da die Visualisierung von räumlichen Flächen per OpenGL oder DirectX zu meist auf Dreiecken basiert (vgl. Abschnitt 2.6.6), sei ferner angenommen, dass die BRep-Flächen des Ausgangszustandes immer als Dreiecke vorliegen. Es wird zu dem vorausgesetzt, dass zu jedem der drei Punkte

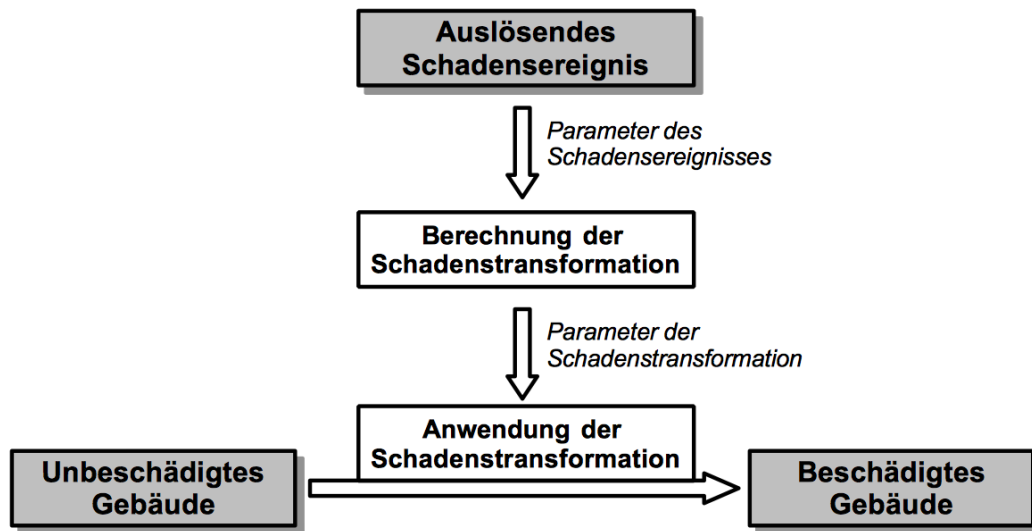


Abbildung 4.1: Prozessmodell zur Erzeugung von beschädigten Gebäuden.

eines Dreiecks eine Farbinformation vorliegt (durch drei Werte im RGB-Farbraum¹). Bildhafte Darstellungen, wie die sogenannten Texturen, werden nicht berücksichtigt und sind Bestandteil von Folgeuntersuchungen und zukünftigen Ansätzen (vgl. Abschnitt 6.1). Zusammengefasst besteht das unbeschädigte Gebäude aus einer Vielzahl räumlicher, farblicher Dreiecke, die die äußere Oberfläche des Gebäudes repräsentieren.

Schadensereignis und Parameter des Schadensereignisses Das *auslösende Schadensereignis* ist ein zur Laufzeit der Simulation auftretendes Geschehen, welches die Erzeugung eines beschädigten Gebäudes zur Folge hat. Das Schadensereignis wird durch ein teilnehmendes Simulationssystem während der Simulation ausgelöst. In einem Simulationssystem wird beispielsweise eine Waffe auf ein Gebäude abgefeuert. Der Beschuss des Gebäudes wird durch eine Vielzahl von Parametern beschrieben. Die Parameter werden als *Parameter des Schadensereignisses* bezeichnet. Mit dem Schadensereignis werden die Parameter des Schadensereignisses instanziiert und vom feuernden Simulationssystem an alle teilnehmenden Simulationssysteme publiziert.

Die *Parameter des Schadensereignisses* beinhalten diejenigen Informationen, die im auslösenden Simulationssystem im Moment des Schadensereignisses zur Verfügung stehen. Je nach Simulationssystem und eingesetztem Waffensystem sind unterschiedliche Informationen zu dem Schadensereignis bekannt, die als Parameter übertragen werden. Dies sind beispielsweise Art und Eigenschaft der Munition, Geschossgeschwindigkeit, Fahr- oder Fluggeschwindigkeit des feuernden Systems oder Weitere.

¹„Der RGB-Farbraum ist ein additiver Farbraum, der Farbwahrnehmungen durch das additive Mischen dreier Grundfarben (**R**ot, **G**rün und **B**lau) nachbildet“ [95].

Berechnung, Parameter und Anwendung der Schadenstransformation Durch ein Schadensereignis wird die Geometrie eines unbeschädigten Gebäudes verändert. Das Gebäude wird beschädigt. Für die Veränderung des Gebäudes werden in dieser Arbeit, wie in Abbildung 4.1 dargestellt, die drei Bezeichnungen *Berechnung der Schadenstransformation*, *Parameter der Schadenstransformation* und *Anwendung der Schadenstransformation* unterschieden.

Die *Berechnung der Schadenstransformation* und die *Anwendung der Schadenstransformation* stellen Berechnungsprozesse dar. Im Schritt der *Berechnung der Schadenstransformation* werden die *Parameter der Schadenstransformation* instanziiert. Die *Parameter der Schadenstransformation* definieren die geometrische Veränderung des Gebäudes. Die Durchführung der Veränderung des Gebäudes erfolgt im Schritt der *Anwendung der Schadenstransformation*.

Die Bezeichnungen werden in einem Beispiel erläutert: Der Schuss einer Handfeuerwaffe führt zu einer Veränderung eines unbeschädigten Gebäudes. Durch den Beschuss entsteht ein Einschussloch. Das Einschussloch könnte exemplarisch durch einen Durchmesser und eine Position am Gebäude definiert werden. Durchmesser und Position sind Beispiele für *Parameter einer Schadenstransformation*, da die Veränderung der Geometrie des unbeschädigten Gebäudes durch diese beiden Parameter definiert wird. Durch den Beschuss entsteht an einer bestimmten Position des Gebäudes ein Einschussloch mit einem bestimmten Durchmesser.

Die Instanziierung der Parameter Position und Durchmesser entspricht dem Schritt der *Berechnung der Schadenstransformation*. Mit der Berechnung der Schadenstransformation könnte beispielsweise der Durchmesser des Einschusslochs aus dem Kaliber der verwendeten Munition abgeleitet werden. Eine mögliche Rechenvorschrift dazu wäre, dass der Durchmesser des Einschusslochs (*Parameter der Schadenstransformation*) immer dem doppelten Kaliber des Projektils (*Parameter des Schadensereignisses*) entspricht. Die Rechenvorschrift dieser *Berechnung der Schadenstransformation* wäre: $D_{Einschussloch} = 2 \cdot D_{Projektile(Kaliber)}$.² Mit dieser Rechenvorschrift würde der Parameter der Schadenstransformation $D_{Einschussloch}$ instanziiert.

Die Geometrie des unbeschädigten Gebäudes muss an der Position des Einschusslochs und mit dem Durchmesser des Einschusslochs entfernt werden, damit das Einschussloch geometrisch entsteht und visualisiert werden kann. Die Herstellung dieser geometrischen Veränderungen am unbeschädigten Gebäude wird als die *Anwendung der Schadenstransformation* bezeichnet.

Beschädigtes Gebäude Das beschädigte Gebäude ist der Endzustand der dargestellten Prozesskette in Abbildung 4.1. Das beschädigte Gebäude repräsentiert die geometrischen Auswirkungen eines Schadensereignisses. Die Geometrie des beschädigten Gebäudes wird, analog zum unbeschädigten Gebäude, durch eine Vielzahl von räumlichen, farblichen Dreiecken repräsentiert.

² D für Durchmesser

4.2 Diskussionen und Herleitungen zur Erzeugung von beschädigten Gebäuden

Die Bestandteile der zuvor dargestellten Prozesskette (vgl. Abbildung 4.1) werden in diesem Abschnitt wieder aufgegriffen. An Hand der Bestandteile dieser Prozesskette wird ein neuer Ansatz zur Erzeugung von Gebäudeschäden hergeleitet.

4.2.1 Parameter des Schadensereignisses

Durch welche Parameter ein Schadensereignis beschrieben wird, hängt davon ab, welche Informationen im Simulationssystem bekannt sind. Bei Handfeuerwaffen (Pistole oder Gewehr) kann beispielsweise die Information zum eingesetzten Kaliber als Parameter übertragen werden, um nachfolgend die Größe eines Einschusslochs berechnen zu können. Bei Schadensereignissen durch explodierende Geschosse ist es hingegen nicht sinnvoll die Größe des Einschusslochs aus dem Kaliber abzuleiten, da für die Beschädigung die Explosion des Geschosses und nicht der ballistische Einschlag maßgeblich ist. Während also bei Handfeuerwaffen Informationen zum Kaliber benötigt werden, sind bei explodierenden Geschossen Kennzahlen zur Explosion von Interesse. Somit unterscheiden sich die Parameter des Schadensereignisses je nach eingesetztem Waffensystem.

Außerhalb dieser Vereinfachung (der Unterscheidung zwischen explosiven und nicht explosiven Geschossen) ist die Auflistung und Gruppierung realer Waffensysteme und Munitionsarten wesentlich umfangreicher (vgl. [77]). Welche Parameter tatsächlich bei Schadensereignissen in einer Simulation publiziert werden, ist stets von den Eigenschaften der teilnehmenden Simulationssysteme und eingesetzten Waffensysteme abhängig.

Diese Abhängigkeit zu einzelnen Simulationssystemen zeigt sich auch darin, dass der Standard des RPR-FOM keine abgeschlossene Parametrisierung von Schadensereignissen implementiert (vgl. Abschnitt 2.4.9). Das RPR-FOM formuliert einen Minimalkonsens mit teils optionalen Parametern (vgl. Tabelle 2.6). Erst wenn in einer Simulation die teilnehmenden Simulationssysteme und somit die teilnehmenden Waffensysteme bekannt sind, werden die konkreten Parameter von Schadensereignissen für die Simulation festgelegt. Das RPR-FOM wird für den Einzelfall erweitert, wenn die teilnehmenden Simulationssysteme mit ihren Eigenschaften feststehen.³

Für einen neuen Ansatz zur Handhabung von Schadensereignissen muss die Anforderung der Integrierbarkeit (vgl. Abschnitt 1.2) berücksichtigt werden. Ein neuer Ansatz darf nicht auf einzelne Simulationen oder spezielle Simulationssysteme beschränkt sein. Die Unabhängigkeit des Ansatzes ist gefordert. Gerade die Parameter von Schadensereignissen sind jedoch, wie zuvor dargestellt wurde, von der einzelnen Simulation abhängig. Durch welche Parameter ein Schadensereignis beschrieben wird, variiert je nach Simulationssystem und eingesetztem Waffensystem.

Der Anforderung der Integrierbarkeit kann nur nachgekommen werden, in dem der neue

³So geschehen im Projekt VIntEL (vgl. Abschnitt 2.7.2).

Ansatz nicht auf konkrete Parameter von Schadensereignissen beschränkt wird. Wenn in einem neuer Ansatz stets spezifische Parameter für Schadensereignisse festgelegt wären, könnte er auch nur in spezifischen Simulationen mit bekannten Simulationssystemen angewendet werden. Diese Anwendbarkeit widerspricht der Anforderung der Integrierbarkeit. Ein neuer Ansatz muss daher, bezüglich der Parameter der Schadensereignisse, generisch sein.

In den nachfolgenden Abschnitten wird ein Verfahren vorgestellt wird, welches keine Abhängigkeit zu konkreten Parametern der Schadensereignisse aufweist. Es wird ein generischer Ansatz beschrieben.

Der neue Ansatz kann in der Praxis nur angewendet werden, in dem er für eine einzelne Simulation konkretisiert wird. Wenn die teilnehmenden Simulationssysteme einer verteilten, virtuellen Simulation bekannt sind, muss der Ansatz für die auftretenden Parameter der Schadensereignisse spezialisiert werden. Das eine solche Spezialisierung problemlos möglich ist, wird prototypisch in Abschnitt 5.1.2 gezeigt.

4.2.2 Berechnung der Schadenstransformation

Die *Berechnung der Schadenstransformation* ist die Rechenvorschrift zur Ableitung von geometrischen Veränderungen aus den Parametern des Schadensereignisses. Für die Berechnung der Schadenstransformation werden Informationen zum Schadensereignis, also die Parameter des Schadensereignisses, benötigt. Die Berechnung der Schadenstransformation ist von den Parametern des Schadensereignisses abhängig (vgl. Abbildung 4.1).

Zuvor wurde in Abschnitt 4.2.1 dargestellt, dass ein neuer Ansatz bezüglich der Parameter der Schadensereignisse generisch sein muss. Die in Abschnitt 4.1 genannte exemplarische Rechenvorschrift kann nur dann angewendet werden, wenn das Kaliber des Projektils als Parameter des Schadensereignisses von einem Simulationssystem publiziert wird. Für die Berechnung des Durchmessers des Einschusslochs ist dieser Parameter zwingend erforderlich. Die Rechenvorschrift ist somit vom Parameter des Kalibers abhängig. Ob der Parameter jedoch publiziert wird, ist wiederum vom einzelnen Simulationssystem abhängig. Erst im Einzelfall einer speziellen Simulation steht fest, durch welche Parameter das Schadensereignis beschrieben wird. Somit ist auch die Rechenvorschrift, die Berechnung der Schadenstransformation, davon abhängig welche Simulationssysteme teilnehmen und welche Waffensysteme eingesetzt werden.

Analog zu den Parameter des Schadensereignisses kann daher für die Berechnung der Schadenstransformation folgendes hergeleitet werden: Damit der neue Ansatz in verschiedenen Simulationen eingesetzt werden kann (wie in Abschnitt 1.2 gefordert wurde), darf der neue Ansatz nicht auf spezifische Berechnungen von Schadenstransformationen beschränkt sein. Die Berechnung der Schadenstransformation muss ebenso generisch sein, wie die Parameter der Schadensereignisse.

Beispiele für Berechnungen von Schadenstransformationen werden in Abschnitt 5.1.2 aufgezeigt.

4.2.3 Parameter der Schadenstransformation

4.2.3.1 Bestehende Ansätze

In den vorherigen Kapiteln wurden bereits Vorgehensweisen zur Erzeugung von beschädigten Gebäuden in der verteilten, virtuellen Simulation beschrieben, die unterschiedliche Parameter der Schadenstransformation verwenden. Diese Vorgehensweisen sind das LoS-Konzept (vgl. Abschnitt 1.1) und die geometrische Veränderung auf Basis geometrischer Primitive (vgl. Abschnitte 2.4.9 und 4.1). In Tabelle 4.1 ist eine Zuordnung dieser beiden Ansätze zu den Bestandteilen der Prozesskette aus Abbildung 4.1 dargestellt. Beide Ansätze werden im Bezug auf die Parameter der Schadenstransformation im Folgenden erläutert.

	Level Of States Konzept	Geometrische Primitive (Beispiel Einschussloch)
Parameter der Schadenstransformation	1 Parameter: Prozentualer Zerstörungsgrad.	2 Parameter: Position und Durchmesser des Einschusslochs.
Anwendung der Schadenstransformation	Einblendung von vorgerechneten und typisiert beschädigten Gebäuden.	Geometrische Verschneidung der geometrischen Primitive mit dem unbeschädigten Gebäude.
Beschädigtes Gebäude	Die beschädigten Gebäude können nur dem typisierten Schadensereignis des vorgerechneten 3D-Modells entsprechen.	Die Gestalt der beschädigten Gebäude ist stets auf die Form der geometrischen Primitive beschränkt. Im Beispiel des Einschusslochs, könnte das unbeschädigte Gebäude nur durch Einschusslöcher verändert werden.

Tabelle 4.1: Vergleich bestehender Vorgehensweisen der Schadenstransformation zur Prozesskette aus Abbildung 4.1

Prozentualer Zerstörungsgrad Auch beim LoS-Konzept wird die Veränderung der Gebäude durch Parameter beschrieben. Durch den prozentualen Zerstörungsgrad wird die Veränderung des Gebäudes definiert. Die vorgerechneten Stufen des 3D-Gebäudemodells werden an Hand des Zerstörungsgrades ein- oder ausgeblendet. Der prozentuale Zerstörungsgrad ist ein Parameter der Schadenstransformation. Beim LoS-Konzept wird somit genau ein Parameter der Schadenstransformation verwendet.

Bewertung: Die Nachteile und Probleme der Parametrisierung mit nur einem Wert sind in den vorangegangenen Abschnitten dargestellt worden. Beispielsweise können die vorgerechneten 3D-Gebäudemodelle des LoS-Konzepts keinesfalls die echten Wechselwirkungen eines Schadensereignisses berücksichtigen. Zur Parametrisierung der Schadenstransformation wird mehr als nur ein Parameter benötigt.

Geometrisch primitive Körper Der zweite Ansatz zur Parameterisierung der Schadenstransformation ist die Definition primitiver Körper. Die Parameter der Schadenstrans-

formation können so gewählt werden, dass ein geometrisch primitiver (3D-) Körper definiert wird. Zur Erzeugung der beschädigten Gebäude wird der definierte Körper mit dem unbeschädigten Gebäude verschnitten, um ein beschädigtes Gebäude zu erzeugen. Ein Beispiel für Parameter der Schadenstransformation, die einen primitiven Körper definieren, sind die Parameter Durchmesser und Position zur Erzeugung von Einschusslöchern (vgl. Abschnitt 4.1). Über die Parameter Durchmesser und Position wird ein Kreisscheibe definiert. Beim Verschneiden der Kreisscheibe mit dem unbeschädigten Gebäude (Anwendung der Schadenstransformation) entsteht das Einschussloch. Die Schadenstransformation zur Erzeugung von Einschusslöchern kann durch zwei Parameter definiert werden.

Auch die in Abschnitt 2.4.9 beschriebenen Einschlagskrater, die durch Position, Radius und Tiefe definiert werden, sind ein Beispiel der Parameterisierung auf Basis primitiver Körper. Durch die Parameter wird ein kegelförmiger Krater definiert. Beim Verschneiden des Kegels mit dem unbeschädigten Gebäude (Anwendung der Schadenstransformation) würde ein kegelförmig beschädigtes Gebäude entstehen (Gebäude mit Einschlagskrater).

Bewertung: Durch die Verwendung primitiver Körper können Wechselwirkungen des Schadensereignisses berücksichtigt werden. Das Einschussloch oder der Einschlagskrater würde genau an der Position entstehen, die durch die Parameter des Schadensereignisses übermittelt würde. Bei der Einblendung vorgerechneter und typisiert beschädigter Gebäude (LoS-Konzept) kann die echte Position des Einschusses, die erst während der Simulation feststeht, nicht berücksichtigt werden. Mit der Definition von primitiven Körpern können somit vielfältigere und authentischere beschädigte Gebäude erzeugt werden, als mit prozentualen Zerstörungsgraden des LoS-Konzepts.

Der Nachteil primitiver Körper ist, dass die erzeugbaren beschädigten Gebäude nur so vielfältig sind, wie es die Parameter der geometrischen Primitive zulassen. Die Anzahl der Parameter limitiert die Vielfältigkeit des beschädigten Gebäudes. Mit den Parametern Position und Durchmesser ist es nur möglich einzelne kreisförmige Einschusslöcher zu definieren.

4.2.3.2 Anforderungen

Im vorherigen Abschnitt wurden die Nachteile der bestehende Ansätze zur Parametrisierung von Schadenstransformationen aufgezeigt. Das die erzeugbaren geometrischen Veränderungen auf konkrete spezielle Formen beschränkt sind, ist ein Nachteil. Ein neuer Ansatz muss gegenüber diesen Nachteilen eine Verbesserung darstellen.

Eine Anforderung eines neuen Ansatzes ist daher, dass durch die Parameter der Schadenstransformation unterschiedliche geometrische Veränderungen definiert und, mit der Anwendung der Schadenstransformation, unterschiedliche beschädigte Gebäude erzeugt werden können (vgl. Anforderung der Authentizität in Abschnitt 1.2). Aus diesen Anforderungen werden zwei geforderte Eigenschaften abgeleitet:

1. **Modifikationsort (Generizität bezüglich des Ortes der geometrischen Veränderungen):** Mit den Parametern der Schadenstransformationen müssen geome-

trische Veränderungen in der Bandbreite von kleinsten Einschusslöchern bis zur vollständigen Zerstörung des Gebäudes definiert werden können. Durch ein Schadenereignis können an unterschiedlichen Position des unbeschädigten Gebäudes verschiedene geometrische Veränderungen auftreten. Dies schließt auch mit ein, dass gleichzeitige Veränderungen an unterschiedlichen Positionen auftreten können (Beispielsweise mehrfache Einschusslöcher durch Splitterwirkungen). Geometrische Veränderungen können an unterschiedlichen Position sowie in unterschiedlicher Größe auftreten und müssen nicht räumlich zusammenhängend sein. Der Modifikationsort muss generisch sein.

- 2. Modifikationsart (Generizität bezüglich der Art der geometrischen Veränderungen):** Im Beispiel des Einschusslochs wurde die geometrische Veränderung räumlich durch Kreisscheiben definiert. Die Art der geometrischen Veränderung wurde jedoch nur implizit beschrieben. Bei Einschusslöchern wurde angenommen, dass bei der Anwendung der Schadenstransformation die Bestandteile der Geometrie des unbeschädigten Gebäudes, die sich innerhalb der definierten Kreisscheibe befinden, entfernt werden, so dass eine Öffnung im Gebäude entsteht. Die Art der geometrischen Veränderung, im Weiteren auch Modifikationsart genannt, ist in diesem Beispiel das *Entfernen* vorhandener Geometrie. Bei Einschusslöchern, wie sie bisher definiert wurden, wurde ausschließlich die Modifikationsart *Entfernen* beschrieben. Neben dieser Modifikationsart sind auch weitere Arten der geometrischen Veränderung möglich. So wird bei dem zuvor beschriebenen Einschlagskrater im Gelände keine Geometrie entfernt, sondern die vorhandene Geometrie so verschoben, dass eine kegelförmige Wölbung im Gelände entsteht. Die Modifikationsart zur Erzeugung von Einschlagskratern ist das *Verschieben*. Die genannten Beispiele, Einschussloch und Einschlagskrater, sind geometrische Veränderungen, die beide bei Waffeneinsätzen zu erwarten sind. Sowohl das *Verschieben*, als auch das *Entfernen* von Bereichen des unbeschädigten Gebäudes sind mögliche Modifikationsarten zur Erzeugung beschädigter Gebäude. Bei einem Einschussloch wird die Geometrie entfernt und bei einem Einschlagskrater wird die Geometrie verschoben. Beide Modifikationsarten werden benötigt. Ein neuer Ansatz darf daher nicht auf eine einzelne Modifikationsart beschränkt sein. Die Modifikationsart muss generisch sein.

Zur Anforderung der Modifikationsart Eine Anforderung an die Parameter der Schadenstransformation ist die Art der Modifikation. Ein neuer Ansatz darf nicht auf eine einzelne Art der Modifikation beschränkt sein. Wie ein neuer Ansatz dieser Anforderung entsprechen kann wird im Folgenden an Hand einer Gebäudesprengung, die in Abbildung 4.2 abgebildet ist, hergeleitet.

Nach der Definition in Abschnitt 3.1 ist ein Gebäudeschaden stets einen stabiler Endzustand. Von der Zeitreihe aus Abbildung 4.2 ist nur das unbeschädigte Gebäude links und



Abbildung 4.2: Zeitreihe einer Gebäudesprengung (Fotomontage der originalen Abbildung aus [95]).

das beschädigte Gebäude äußerst rechts von Interesse. Mit den Parametern der Schadenstransformation muss nur der Übergang zwischen diesen zwei Zuständen definiert werden. Zwischenstände, wie sie in Abbildung 4.2 dargestellt sind, werden nicht benötigt.

Aus der Perspektive eines Beobachters, genauso wie sie der Übende in der virtuellen Simulation hat, kann die geometrische Veränderung zwischen diesen beiden Zuständen durch zwei Modifikationsarten approximiert werden. Entweder es werden Teile der Bausubstanz so starken Kräften ausgesetzt, dass sie pulverisiert werden und Staub entsteht (*Modifikation Entfernen*), oder sie erfahren einen geometrischen Positionswechsel (*Modifikation Verschieben*).

In Abbildung 4.3 sind diese beiden Modifikationsarten *Verschieben* und *Entfernen* in einer Fotomontage abgebildet. Auf das unbeschädigte Gebäude wird eine Schadenstransformation mit zwei Modifikationsarten angewendet. Der obere Teil des Schornsteins wird auf den Boden *verschoben*, während der übrige Teil des Schornsteins *entfernt* wird. Durch die Anwendung der Schadenstransformation mit diesen beiden Modifikationen entsteht das beschädigte Gebäude.

Die Anwendung der Schadenstransformation ist in diesem Beispiel nicht auf eine einzelne Modifikationsart beschränkt. Der Schornstein wird durch zwei Modifikationsarten verändert. Die Anwendung von zwei Modifikationsarten wird realisiert, in dem jeder Modifikationsart ein Wirkungsbereich zugeordnet wird. Die Modifikation *Verschieben* wirkt auf den oberen Bereich des Schornsteins und die Modifikation *Entfernen* wirkt auf den unteren Bereich des Schornsteins.

Der geforderten Generizität der Modifikationsart kann somit entsprochen werden, in dem jede Modifikation räumlich durch einen Wirkungsbereich beschränkt wird. Auf ein unbeschädigtes Gebäude können dadurch unterschiedliche geometrische Veränderungen wirken. Die Größe und die Position des Wirkungsbereichs (beim *Verschieben* und beim *Entfernen*) sowie die Richtung und Länge der Verschiebungsmodifikation sind durch die Parameter der Schadenstransformation genauer zu definieren.

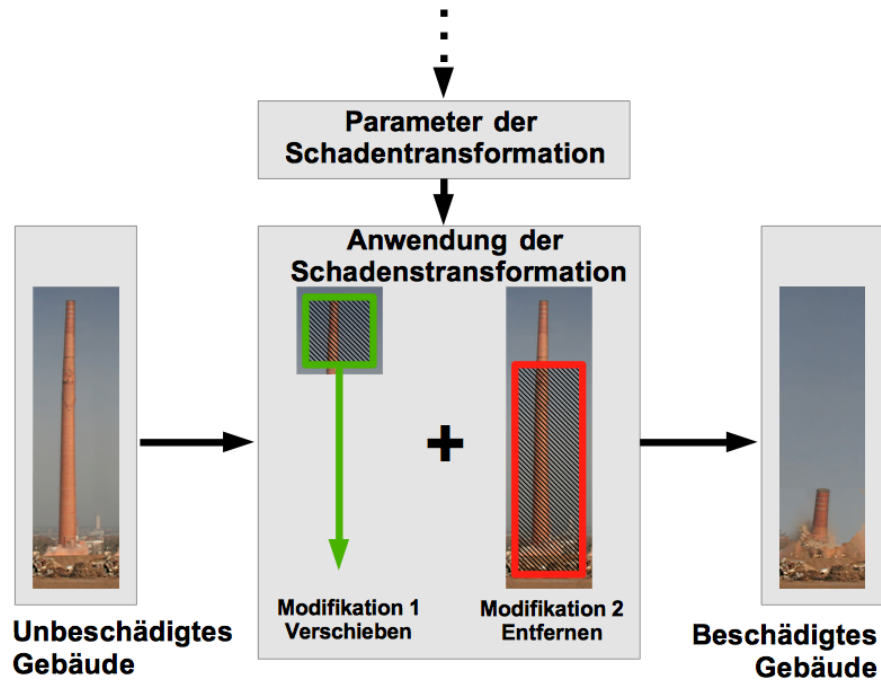


Abbildung 4.3: Anwendung der Schadenstransformation am Beispiel der Modifikationsarten *Verschieben* und *Entfernen*, die in den Parametern der Schadenstransformation definiert werden (Fotomontage, Schornsteinabbildung aus [95]).

Zur Anforderung des Modifikationsortes Eine weitere Anforderung an die Parameter der Schadenstransformation ist der Ort der Modifikation. Geometrische Modifikationen müssen in unterschiedlicher Größe und an unterschiedlichen Positionen des Gebäudes definiert werden können. Ferner folgt aus dem vorherigen Abschnitt, dass durch die Parameter der Schadenstransformation sogenannte Wirkungsbereiche definiert werden müssen, um unterschiedliche Modifikationsarten anwenden zu können (Anforderung der Modifikationsart). Wie ein neuer Ansatz die Anforderungen des Modifikationsortes berücksichtigen kann wird im Folgenden hergeleitet.

Es ist davon auszugehen, dass in jedem Sichtsystem eines Simulationssystems bildhafte Informationen zur Darstellung der synthetischen Umwelt verwendet werden (vgl. Abbildung 2.2). Beispielsweise wird auf dem 3D-Geländemodell zu meist ein Luftbild visualisiert (vgl. Abbildung 2.2). Die Datengrundlage für diese Visualisierungen sind digitale Bilder also Rastergrafiken. Die Auflösung dieser Digitalbilder, also die Anzahl der Pixel in einem Bild, ist stets beschränkt. Das Luftbild ist beispielsweise in einer bestimmten Auflösung aufgenommen worden und wird mit einer bestimmten Auflösung visualisiert. Die Auflösung der Rastergrafiken in einem Sichtsystem ist stets beschränkt.

Welche Auflösung benötigt wird, ist vom Simulationssystem abhängig. Für ein Simulationssystem eines Flugzeugs könnte eine Bodenaufösung des Luftbildes von 10 cm ausreichend sein. Wenn das Flugzeug während der Simulation keine Position einnehmen kann, in der das Luftbild mit einer größeren Auflösung als den 10 cm visualisiert wird, ist die Bodenauf-

lösung von 10 cm ausreichend. Eine detaillierte Darstellung des Luftbildes im Subdezimeterbereich wird nicht benötigt. Die Auflösung des Luftbildes für dieses Simulationssystem kann auf 10 cm beschränkt werden.

Diese Eigenschaft, dass die Auflösung von Rastergrafiken stets beschränkt ist, kann auf die Parametrisierung der Schadenstransformation übertragen werden. Bisher geht mit der Anforderung bezüglich Modifikationsort einher, dass Modifikationen in *unterschiedlicher Größe* und an *unterschiedlichen Position* definiert werden müssen. Wie unterschiedlich groß und an welchen unterschiedlichen Positionen eine Modifikation auftritt, kann ebenso beschränkt werden, wie die Auflösung von Rastergrafiken.

Eine geometrische Veränderung muss nur so detailliert berechnet (Berechnung der Schadenstransformation), definiert (Parameter der Schadenstransformation) und angewendet werden (Anwendung der Schadenstransformation), wie sie in den Simulationssystemen benötigt wird. Da die Simulationssysteme bemannter Waffensysteme in der synthetischen Umwelt zu meist außerhalb von Gebäuden agieren (vgl. Abschnitt 2.3.3), kann (analog zur Bodenauflösung) eine Beschränkung der Auflösung der Gebäudegeometrie angenommen werden. Nicht nur die Pixel eines Luftbildes, sondern auch die Geometrie eines 3D-Gebäudemodells wird nur bis zu einer maximalen Auflösung benötigt. Geometrische Veränderungen die so klein sind, dass sie im Simulationssystem niemals visualisiert werden, müssen nicht berechnet, definiert und angewendet werden.

Mit der Parametrisierung der Schadenstransformationen müssen zwar geometrische Modifikationen in unterschiedlicher Größe und an unterschiedlichen Position des Gebäudes definiert werden können, jedoch nur bis zu einer bestimmten Auflösung.

Analog zu typischen Luftbildern wird diese benötigte Auflösung zur Herleitung eines neuen Ansatzes mit 10 cm angenommen.⁴ Es werden zunächst nur geometrische Veränderungen betrachtet, die größer sind als 10 cm. Im letzten Kapitel dieser Arbeit werden auch Gebäudeschäden bei variierender Auflösung erzeugt (vgl. Abschnitt 5.2.2).

4.2.3.3 Neuer Ansatz

In diesem Abschnitt wird eine Parametrisierung der Schadenstransformation als Ansatz dieser Arbeit hergeleitet. In den beiden vorherigen Abschnitten wurde zum einen die Notwendigkeit der Wirkungsbereiche und zum anderen die Beschränkung der Auflösung aufgezeigt. Beides wird in der nachfolgenden Herleitung aufgegriffen.

Durch eine Beschränkung der benötigten Auflösung, auf beispielsweise 10 cm, ist es nicht notwendig, dass die Parameter der Schadenstransformation geometrische Veränderungen in **jeder** Größe und an **jeder** Position definieren. Nur geometrische Veränderungen mit einer Mindestgröße von 10 cm müssen durch die Parameter der Schadenstransformation definiert werden. Aus dieser Mindestgröße folgt, dass auch die Wirkungsbereiche der Modifikationen nicht kleiner als 10 cm sein müssen.

⁴Herleitung des Wertes: 10cm ist die Luftbildauflösung, die die Luftbilder auf den einschlägigen Geoportalen der deutschen Landes- und Kommunalverwaltungen aufweisen.

Eine Auflösung von 10 cm, beispielsweise bei Luftbildern, bedeutet, dass ein Pixel auf der Erdoberfläche eine Fläche von 1 dm^2 abdeckt. An der dreidimensionalen Geometrie eines 3D-Gebäudes kann die Auflösung von 10 cm als Volumen interpretiert werden. Bei einer Auflösung von 10 cm entspricht das kleinste relevante Volumen einem dreidimensionalen Würfel mit einer Kantenlänge von 10 cm, also einem Volumen von 1 dm^3 . Alle geometrischen Veränderungen, die kleiner sind als ein 1 dm^3 , sind nicht relevant.

Aus der Anforderung der Modifikationsart folgt, dass eine geometrische Veränderung nicht auf eine einzelne Modifikation beschränkt sein darf. Eine geometrische Veränderung kann aus mehreren Modifikationen bestehen. Jede einzelne Modifikation hat einen Wirkungsbereich, der größer oder gleich 1 dm^3 ist. Diese Annahme, dass eine geometrische Veränderung aus vielen Modifikationen besteht, kann als Ansatz zur Parametrisierung der Schadenstransformation verwendet werden. Die Aussage, *eine geometrische Veränderung besteht aus einer Vielzahl von Modifikationen mit einem jeweiligen Wirkungsbereich von 1 dm^3* , gilt nicht nur für **eine** geometrische Veränderung. Die Aussage kann für die Gesamtheit aller relevanten geometrischen Veränderungen erweitert werden: **Jede** geometrische Veränderung besteht aus einer Vielzahl von Modifikationen. Mit anderen Worten: Durch eine Vielzahl von Modifikationen mit einem jeweiligen Wirkungsbereich von 1 dm^3 , kann jede relevante geometrische Veränderung definiert werden. Daraus folgt eine mögliche Parametrisierung der Schadenstransformation: Die Parameter der Schadenstransformation bestehen stets aus einer Vielzahl von Modifikationen mit einem jeweiligen Wirkungsbereich von 1 dm^3 .

Analog zur Abbildung 4.3 kann eine einzelne Modifikation auch als Vektor interpretiert werden. Ein Vektor hat einen Wirkungsbereich (vgl. Abbildung 4.3 oberer Teil des Schornsteins), eine Richtung und eine Länge (beispielsweise Verschiebung bis auf Bodenniveau). Diese Vektoren werden im Weiteren auch als Modifikationsvektoren bezeichnet. Ein Modifikationsvektor kann, auch ohne Länge und Richtung, nur aus einer Modifikationsart und einem Wirkungsbereich bestehen, um beispielsweise das Entfernen des unteren Bereiches des Schornsteins zu definieren (vgl. Abbildung 4.3).

Alle Modifikationen eines einzelnen Schadensereignisses, also alle Modifikationsvektoren, können wiederum als Vektorfeld interpretiert werden. Das Vektorfeld wird auch als Modifikationsvektorfeld bezeichnet. Im Beispiel aus Abbildung 4.3 besteht das Modifikationsvektorfeld aus zwei Modifikationsvektoren: Ein Modifikationsvektor mit der Modifikationsart *Entfernen* und ein Modifikationsvektor mit der Modifikationsart *Verschieben*. Das Modifikationsvektorfeld mit den beiden Modifikationsvektoren definiert die geometrische Veränderung des Gebäudes. **Das Modifikationsvektorfeld entspricht den Parametern der Schadenstransformation.**

4.2.3.4 Erläuterungen zum Ansatz

Die Interpretation der Parameter der Schadenstransformation als Modifikationsvektorfeld wird als Ansatz dieser Arbeit verwendet. Die Parameter der Schadenstransformation de-

finieren stets eine Vielzahl von Modifikationsvektoren. Jeder einzelne Vektor stellt eine geometrische Modifikation dar und bewirkt eine geometrische Veränderung für seinen Wirkungsbereich.

Das Modifikationsvektorfeld zum Beispiel in Abbildung 4.3 besteht aus zwei Modifikationsvektoren. Die kleine Anzahl von Modifikationen ist möglich, da innerhalb eines Wirkungsbereichs (oberer Teil des Schornsteins und unterer Teil des Schornsteins) gleiche Veränderungen ausgeführt werden. Die Geometrie innerhalb eines Wirkungsbereichs wird homogen modifiziert. Die Geometrie, die sich innerhalb des oberen Bereichs befindet, wird homogen (gleiche Richtung und gleicher Betrag) verschoben und die Geometrie, die sich innerhalb des unteren Bereichs befindet, wird entfernt.

Außerhalb dieses vereinfachenden Beispiels muss davon ausgegangen werden, dass durch Schadensereignisse zu meist nicht homogene Modifikationen entstehen. Wenn, beispielsweise nach einer Explosion, die Bruchstücke eines Gebäudes radial um den Explosionsort versprengt werden, dann wird jedes Bruchstück in eine andere Richtung und um einen anderen Betrag *verschoben*. Auf jedes Bruchstück wirkt ein separater Modifikationsvektor. Das Modifikationsvektorfeld besteht aus einer Vielzahl unterschiedlicher Vektoren. Das Modifikationsvektorfeld ist nicht homogen.

Gleichwohl kann die Anzahl der maximal benötigten Modifikationsvektoren über die Mindestgröße beschränkt werden. Jeder Wirkungsbereich hat eine Mindestgröße von 1 dm^3 . Wenn auf jeden Kubikdezimeter des 3D-Gebäudemodells eine Modifikation angewendet werden müsste (entspricht einem Totalschaden des Gebäudes), ergibt sich die Anzahl der benötigten Modifikationen aus der Größe des Gebäudes. Um diese Anzahl der möglichen Modifikationen beschränken zu können (vgl. nachfolgender Abschnitt), wird neben der Mindestgröße eines einzelnen Wirkungsbereichs (10 cm) auch eine Maximalgröße für Gebäude festgelegt. Es sei exemplarisch angenommen, dass typische Gebäude maximal 10 m breit, 10 m hoch und 10 m tief sind. Ein Gebäude mit den Maßen $10 \cdot 10 \cdot 10$ Metern besteht umgerechnet aus $100 \cdot 100 \cdot 100 \text{ dm}^3$. Dann ist entsprechend anzunehmen, dass ein Modifikationsvektorfeld maximal $100^3 = 1.000.000$ Modifikationsvektoren enthält.

Die konkrete Anzahl der Modifikationsvektoren ist von den Parametern des Schadensereignisses abhängig. Wenn beispielsweise eine leichte Explosion mit etwas Abstand zum Gebäude auftritt, wird die geometrische Veränderung am Gebäude kleiner sein als eine starke Explosion unmittelbar am Gebäude. Bei kleinen geometrischen Veränderungen werden entsprechend weniger Modifikationsvektoren benötigt als bei großen geometrischen Veränderungen. Demnach stellt ein leeres Modifikationsvektorfeld die kleinste mögliche geometrische Veränderung dar und das voll besetzte Modifikationsvektorfeld die größte geometrische Veränderung. Es ist offensichtlich, dass das leere Modifikationsvektorfeld auch keine geometrischen Veränderungen verursacht. Das Gebäude bleibt unverändert. Die Anwendung der Schadenstransformation findet nicht statt. Ein voll besetztes Modifikationsvektorfeld führt dagegen zu einer Anwendung der Schadenstransformation, die den Extremfall der größten möglichen geometrischen Veränderung darstellt. Da bei einem voll besetzten Modifikationsvektorfeld jeder Kubikdezimeter des Gebäudes verändert wird, stellt auch der

rechnerische Aufwand der Anwendung der Schadenstransformation einen Extremfall dar. Dies führt auf folgende Annahme: Wenn ein voll besetztes Modifikationsvektorfeld der Anforderung der Reaktionszeit genügt (vgl. Abschnitt 1.2), werden auch kleinere geometrische Veränderungen, mit einem nicht voll besetzten Modifikationsvektorfeld, innerhalb der geforderten Reaktionszeit berechnet und visualisiert werden können. Aus diesem Grund wird für den weiteren Verlauf dieser Arbeit angenommen, dass das Modifikationsvektorfeld voll besetzt ist. In Kombination zu den vorherigen Annahmen bezüglich der Größe des Gebäudes wird folgendes angenommen: Ein Modifikationsvektorfeld besteht stets aus 1.000.000 Modifikationsvektoren.

Der Wirkungsbereich eines Modifikationsvektors wurde zuvor bereits als Voxel bezeichnet. Durch die Annahmen zur typischen Gebäudegröße und zur maximalen Auflösung können die Wirkungsbereiche des Modifikationsvektorfeldes auch als Voxelmodell beschrieben werden. Die Wirkungsbereiche eines Modifikationsvektorfeldes stellen zusammen genommen ein $10 \cdot 10 \cdot 10$ Meter großes Voxelmodell dar, welches aus $100 \cdot 100 \cdot 100$ Voxeln besteht. Jedes Voxel hat die Form eines Würfels mit der Kantenlänge von 10 cm.

4.2.4 Anwendung der Schadenstransformation

Im vorherigen Abschnitt wurde das Modifikationsvektorfeld als Parameterisierung der Schadenstransformation vorgestellt. In diesem Abschnitt wird erläutert, wie ein Modifikationsvektorfeld auf das unbeschädigte Gebäude angewendet werden kann. Ein Ansatz für die Anwendung der Schadenstransformation wird erläutert. Für die Anwendung der Schadenstransformation sind drei Annahmen aus den vorherigen Abschnitten relevant:

1. Das unbeschädigte Gebäude besteht aus einer Vielzahl von Dreiecken (vgl. Abschnitt 4.1).
2. Das Modifikationsvektorfeld besteht aus 1 Mio. Modifikationen, die jeweils einen Wirkungsbereich von 1 dm^3 haben (vgl. vorheriger Abschnitt).
3. Das beschädigte Gebäude besteht ebenfalls aus einer Vielzahl von Dreiecken (vgl. Abschnitt 4.1).

Mit der Anwendung der Schadenstransformation wird das beschädigte Gebäude erzeugt. Das Modifikationsvektorfeld wird auf das unbeschädigte Gebäude angewendet. Die Wirkungsbereiche des Modifikationsvektorfeldes müssen den entsprechenden Dreiecken zugeordnet werden, um diese modifizieren zu können. Die Dreiecke des unbeschädigten Gebäudes und die Wirkungsbereiche der Modifikationen werden verschnitten.

Dreiecksbasierter Ansatz zur Anwendung der Schadenstransformation Es sei als Beispiel angenommen, dass sich ein Dreieck mit drei Wirkungsbereichen überschneidet und die Modifikationen nicht homogen sind. Ferner sei angenommen: Die Geometrie innerhalb des ersten Wirkungsbereiches soll verschoben werden, die Geometrie im zweiten Wirkungsbereich soll unverändert bleiben und die Geometrie im dritten Wirkungsbereich

soll entfernt werden. Die Modifikationen für das Dreieck sind also nicht homogen, weshalb das Dreieck nicht als ganzes modifiziert werden kann. Ein Teil des Dreiecks soll verschoben werden, ein Teil soll unverändert bleiben und ein Teil des Dreiecks soll entfernt werden. Um die drei Teile separat modifizieren zu können, muss das Dreieck an den Grenzen der Wirkungsbereiche zerlegt werden. Mit der Zerlegung wird das Dreieck mit den drei Würfeln (Wirkungsbereichen) verschneiden. Durch die Verschneidung entstehen drei resultierende Flächen, die jeweils exakt innerhalb eines Wirkungsbereiches liegen. Anschließend können die Modifikation auf die Flächen, die durch die Verschneidung entstanden sind, angewendet werden. Die erste Fläche wird verschoben, die zweite Fläche bleibt unverändert und die dritte Fläche wird entfernt. Dieses Vorgehen entspricht der Anwendung der Schadenstransformation.

Aus Abschnitt 4.1 folgt, dass auch das beschädigte Gebäude aus Dreiecken besteht. Aus diesem Grund müssen die Flächen, die bei der Verschneidung entstehen, trianguliert werden. Die Schnittfläche aus einem Würfel (Wirkungsbereich) und einem Dreieck kann aus bis zu sechs Seiten bestehen⁵. Mit der Anwendung der Schadenstransformation muss eine Triangulierung dieser Flächen durchgeführt werden, damit das resultierende beschädigte Gebäude wiederum nur aus Dreiecken besteht.

Es ist davon auszugehen, dass ein Dreieck des unbeschädigten Gebäudes stets in mehr als nur einem Wirkungsbereich des Modifikationsvektorfeldes liegt⁶. Daraus folgt, dass der zuvor skizzierte Schritt der Verschneidung für jeden Wirkungsbereich durchgeführt werden muss, der sich mit einem Dreieck überschneidet. Erst nach diesem Schritten kann die eigentliche Modifikation (Beispielsweise das Verschieben oder das Entfernen der Flächen) durchgeführt werden. Anschließend erfolgt die Triangulierung der Flächen der Verschneidung.

Bewertung des Ansatzes: Grundsätzlich scheint dieser Ansatz zur Anwendung der Schadenstransformation möglich. Es ist jedoch zu vermuten, dass der dreiecksbasierte Ansatz zur Anwendung der Schadenstransformation rechenintensiv ist. Bisher wurde nicht erwähnt, dass sich innerhalb eines Wirkungsbereiches auch mehr als ein Dreieck befinden kann. Bei der Verschneidung müssen alle Dreiecke berücksichtigt werden, die sich innerhalb eines Wirkungsbereiches befinden. Der Prozess *Verschneidung von Wirkungsbereich mit Dreieck* muss unter Umständen für einen Wirkungsbereich wiederholt durchgeführt werden, was weiteren Rechenaufwand zur Folge hat.

Ob eine Implementierung dieses Ansatzes mit der Anforderung der Reaktionszeit (vgl. Abschnitt 1.1) vereinbar ist, kann nicht ohne weitere Untersuchungen bewertet werden. Wie rechenintensiv der Ansatz ist, müsste durch prototypische Implementierungen untersucht werden. Diese Untersuchungen bleiben jedoch zukünftigen Arbeiten vorbehalten. Für diese

⁵Das schneidende Dreieck kann so diagonal im Raum liegen, dass jede Seite des Würfels vom Dreieck geschnitten wird. Da jeder Würfel sechs Seiten kann, kann je nach Lage des Dreiecks, eine Schnittfläche mit sechs Seiten entstehen.

⁶Gebäudeteile werden zu meist durch großflächige Dreiecke modelliert. Wände, Fenster, Türen, Dächer oder Mauern stellen in der Regel ebene Flächen dar. Es ist davon auszugehen, dass die Dreiecke, die diese großflächige Geometrie approximieren, größer sind als die Wirkungsbereiche. Ein Dreieck des unbeschädigten Gebäudes überlappt sich stets mit mehreren Wirkungsbereichen.

Arbeit wird der Ansatz der dreiecksbasierten Anwendung der Schadenstransformation, wie nachfolgend begründet wird, nicht weiter verfolgt.

Voxelbasierter Ansatz zur Anwendung der Schadenstransformation Der Ansatz der dreiecksbasierten Anwendung der Schadenstransformation, wird in dieser Arbeit nicht verwendet. Die Rechenzeit der komplexen Verschneidungs- und Triangulierungsfunktion sind vermutlich nicht mit der Anforderung der Reaktionszeit vereinbar. Diese Vermutung ist jedoch nicht das Hauptargument gegen den Ansatz. Das wesentliche Argument gegen den dreiecksbasierten Ansatz ist, dass die Verschneidungs- und Triangulierungsvorgänge innerhalb eines Wirkungsbereiches ausgeführt werden. Die Größe der Wirkungsbereiche wurde an Hand der maximal benötigten Auflösung von 10 cm hergeleitet. Verschneidung und Triangulierung sind jedoch Vorgänge, die im Subdezimeterbereich ausgeführt werden. Es sind geometrische Modifikationen, die kleiner sind als ein Kubikdezimeter. Selbst wenn der dreiecksbasierte Ansatz zur Anwendung der Schadenstransformation die Anforderung der Reaktionszeit einhalten würde, so sind die damit einhergehenden Verschneidungsoperationen unnötig. Es würde eine Vielzahl von Dreiecken entstehen, die deutlich kleiner wären als ein Kubikdezimeter.

Zuvor wurde dargestellt, dass bei Rastergrafiken, bei einer maximal benötigten Auflösung, das Pixel die kleinste noch benötigte Information ist. Bei einer maximalen Bodenauflösung, beispielsweise bei Luftbildern von 10 cm, wird pro Pixel nur eine Farbinformation benötigt. Es werden keine Informationen verwendet, die kleiner sind als ein Pixel. Dies kann auf die geometrischen Veränderung im Dreidimensionalen übertragen werden. Da ein Kubikdezimeter das kleinste relevante Volumen für geometrische Veränderungen darstellt, sind für die Anwendung der Schadenstransformationen auch nur die Informationen relevant, die größer oder gleich diesem Volumen sind. Alle geometrischen Informationen, die kleiner sind als ein Kubikdezimeter müssen nicht berücksichtigt werden. Pro Voxel, also pro Kubikdezimeter, wird ausschließlich die Information benötigt, ob dort Geometrie vorhanden ist oder nicht und wenn sie vorhanden ist, welche Farbe sie hat.

Wenn nur die Bestandteile der Geometrie relevant sind, die größer sind als ein Kubikdezimeter, dann gilt dies auch für das unbeschädigte Gebäude. Auch vom unbeschädigten Gebäude wird für jeden Kubikdezimeter des 3D-Gebäudemodells nur die Information benötigt, ob dort Geometrie vorhanden ist und wenn ja, welche Farbe sie hat.

Aus diesem Bedarf leitet sich der voxelbasierte Ansatz zur Anwendung der Schadenstransformation ab. Der voxelbasierte Ansatz ist die Idee, bereits vom unbeschädigten Gebäude nur die Geometrie zu verwenden, die für die Anwendung der Schadenstransformation benötigt wird. Das unbeschädigte Gebäude wird in die Struktur überführt, die maximal benötigt wird. Benötigt wird eine Information pro Kubikdezimeter. Mit anderen Worten: Es ist ausreichend, das unbeschädigte Gebäude kubikdezimeterweise zu betrachten, da nur für jeden Kubikdezimeter eine Information benötigt wird. Dazu werden die Dreiecke des unbeschädigten Gebäudes in eine Voxelstruktur überführt, dieser Schritt wird als Voxelisierung bezeichnet (vgl. nachfolgender Abschnitt).

Wenn das unbeschädigte Gebäude nach einer Voxelisierung als Voxelstruktur vorliegt, ist die eigentliche Anwendung der Schadenstransformation, also die Modifikation, trivial. Da für die Wirkungsbereiche die gleiche maximale Auflösung wie für die Voxelisierung von beispielsweise 10 cm verwendet wird, ist keine Verschneidung notwendig. Die Voxel des unbeschädigten Gebäudes sind deckungsgleich mit den Wirkungsbereichen des Modifikationsvektorfeldes. Mit der Anwendung der Schadenstransformation müssen ausschließlich Voxel modifiziert werden.

Der Schritt der Triangulierung ist beim voxelbasierten Ansatz ebenso notwendig, wie beim dreiecksbasierten Ansatz. Während beim dreiecksbasierten Ansatz jedoch sehr kleine Dreiecke entstehen, werden bei diesem Ansatz die Voxel trianguliert, so dass zu meist Dreiecke entstehen, deren Seitenlänge nicht kleiner als 10 cm ist. Dieser Schritt wird als Retriangulierung bezeichnet und ebenfalls im nachfolgenden Abschnitt dargestellt.

4.3 Zusammenfassung - Ein neues Verfahren

Aus den Herleitungen und Diskussionen des vorherigen Abschnitts kann der neue Ansatz zur Handhabung von Gebäudeschäden als Prozesskette dargestellt werden. Die Bestandteile der Prozesskette werden in diesem Abschnitt beschrieben. Die einzelnen Komponenten werden in Abbildung 4.4 dargestellt.

4.3.1 Voxelisierung

Bei der Anwendung der Schadenstransformation wirken die Modifikationsvektoren des Modifikationsvektorfeldes auf diskrete Bereiche des unbeschädigten Gebäudes, den sogenannten *Wirkungsbereichen*. Ein Wirkungsbereich entspricht der Geometrie eines Würfels. Alle Dreiecke oder Bestandteile der Dreiecke des unbeschädigten Gebäudes, die sich innerhalb des Würfels befinden, werden mit dem identischen Modifikationsvektor verändert (entfernt, verschoben oder nicht modifiziert). Für jeden Modifikationsvektor muss zuvor berechnet werden, ob sich Dreiecksbestandteile des Ausgangszustandes innerhalb des Wirkungsbereiches befinden, also ob der Wirkungsbereich leer ist oder nicht. Dazu wird das unbeschädigte Gebäude in ein Voxelmodell transformiert, welches exakt der Auflösung des Modifikationsvektorfeldes entspricht. Dadurch kann jeder Modifikationsvektor eindeutig auf ein Voxel angewendet werden. Die Transformation des Ausgangszustandes in Voxel wird als *Voxelisierung* bezeichnet.

Das Prinzip der Voxelisierung ist in Abbildung 4.5 dargestellt. In dem unbeschädigten Gebäudemodell (links) werden die Zuordnungen zu den Wirkungsbereichen der Modifikationsvektoren definiert. In Abbildung 4.5 (Mitte) ist dies durch die Überlagerung eines VoxelSpace Gittermodells mit dem unbeschädigten Gebäude dargestellt. Jede Gitterzelle entspricht einem Voxel und somit einem Wirkungsbereich eines Modifikationsvektors. Im weiteren Verlauf der Voxelisierung werden für jede Gitterzelle, die nicht leer ist, die Geo-

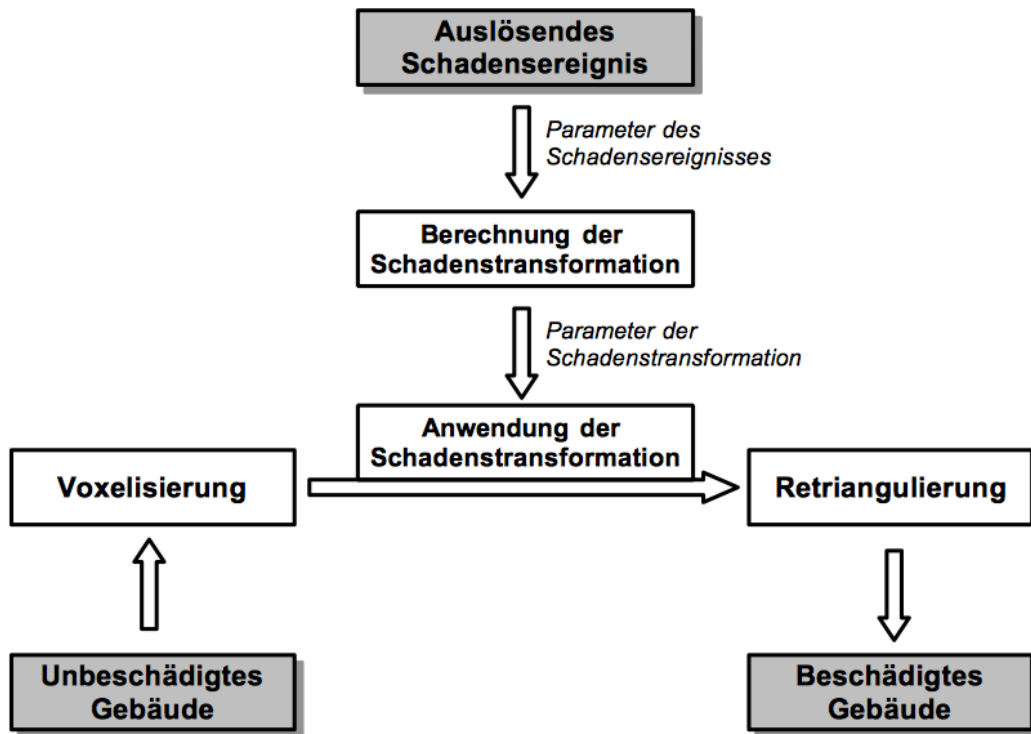


Abbildung 4.4: Erweitertes Prozessmodell zur Handhabung von Gebäudeschäden.

metrie des Zellenmittelpunktes und die Farbe gespeichert (rechts). Im Ergebnis entsteht für jede Zelle eine punktartige Repräsentation.

Grundsätzlich kann die Voxelisierung durch zwei Arten von Schleifen implementiert werden. Die erste Lösung ist eine Iteration über alle Dreiecke des Ausgangszustandes. Für jedes Dreieck werden die entsprechenden Voxel erzeugt. Die zweite Lösung ist eine Iteration über alle Voxel, also alle Wirkungsbereiche des Modifikationsvektorfeldes. Die erste Lösung entspricht einer Iteration über alle Dreiecke und die zweite Lösung einer Iteration über den Raum des Vektorfeldes.

Wenn die Anzahl der Dreiecke geringer ist als die Anzahl der Voxel des Modifikationsvektorfeldes, benötigt die Schleife über die Dreiecke weniger Iterationsschritte als eine Schleife über die Voxel des Modifikationsvektorfeldes. In der Praxis führt dies jedoch zu dem Effekt, dass die Laufzeit der Voxelisierung von der Anzahl der Dreiecke des Ausgangszustandes abhängig ist. Die Iteration über die Voxel wäre zwar langsamer, dafür jedoch unabhängig von der Komplexität des unbeschädigten Gebäudes. Aufgrund dieser Laufzeitkonstanz, wird der Ansatz der Voxelisierung des Raumes verwendet. Schwarz und Seidel [68] zeigen außerdem, dass die Voxelisierung des Raumes, bei Ausnutzung von Grafikprozessoren (Grafikkarten), performanter sein kann als die herkömmliche Voxelisierung per Iteration über die Dreiecke.

Wenn die Anzahl der Voxel geringer ist als die Anzahl der Dreiecke, sind für die Iteration

über die Voxel ohnehin weniger Berechnungsschritte erforderlich als für die Iteration über die Dreiecke. Auch dann ist der Ansatz der Voxelisierung des Raumes zur Optimierung der Laufzeit zu favorisieren.

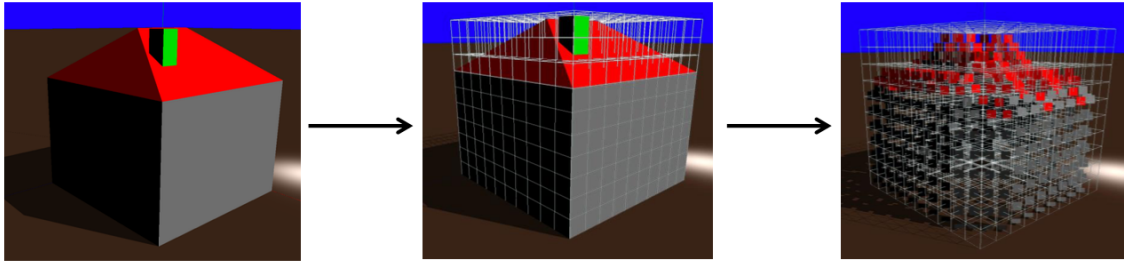


Abbildung 4.5: Die Voxelisierung: Überführung des unbeschädigten Gebäudes in die Voxelstruktur.

4.3.2 Anwendung der Schadenstransformation

Die Anwendung der Schadenstransformation ist der Prozess, bei dem das voxelisierte unbeschädigte Gebäude durch das Modifikationsvektorfeld verändert wird. Die Modifikation ist am Beispiel einer Verschiebung in Abbildung 4.6 dargestellt. Die geometrischen Veränderungen, also die Verschiebungsvektoren, sind in Abbildung 4.6 so gewählt, dass die Funktionsweise der Anwendung der Schadenstransformation deutlich wird. Das Modifikationsvektorfeld entspricht keiner realen Waffenwirkung. Das Modifikationsvektorfeld enthält ausschließlich parallele, jedoch pro Gebäudehälfte entgegengesetzte Verschiebungen. Das 3D-Gebäudemodell wird in zwei Hälften geteilt. Die linke Hälfte des 3D-Gebäudemodells wird nach links gezogen und die rechte Hälfte nach rechts.

In Abbildung 4.6 wurde zur Verdeutlichung der Funktionsweise auch eine reduzierte Auflösung von 1 m gewählt. Die Parameter der Schadenstransformation beinhalten nur $10 \cdot 10 \cdot 10 = 1000$ Modifikationsvektoren, mit einem jeweiligen Wirkungsbereich von 1 m^3 .

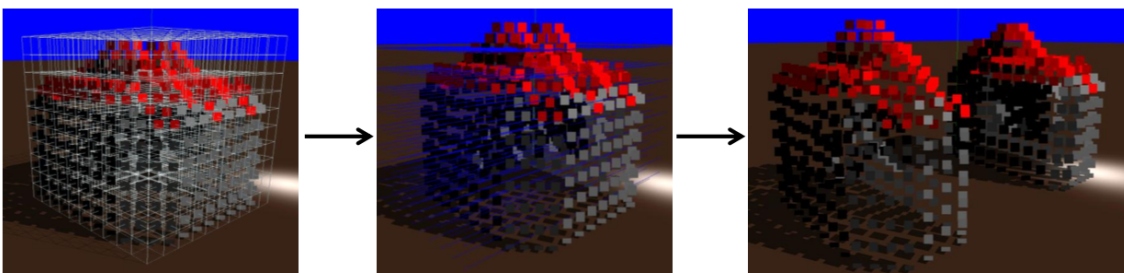


Abbildung 4.6: Anwendung der Schadenstransformation

4.3.3 Retriangulierung

Das Ergebnis des vorherigen Schrittes der Anwendung der Schadenstransformation sind modifizierte Voxel. Für das beschädigte Gebäude wird jedoch eine dreiecksbasierte Repräsentation benötigt. Der Schritt der *Retriangulierung* beinhaltet den Transformationsvorgang der modifizierten voxelbasierten Repräsentation zurück zu einer dreiecksbasierten Repräsentation. Diese dreiecksbasierte Repräsentation ist das beschädigte Gebäude. Den obigen Beispielen folgend, ist in Abbildung 4.7 die Retriangulierung nach einer Verschiebung dargestellt. Die geringe optischen Qualität des beschädigten Gebäude in Abbildung 4.7 ist auf die reduzierte Auflösung zurückzuführen, die zur Veranschaulichung der Ansatzes gewählt wurde.

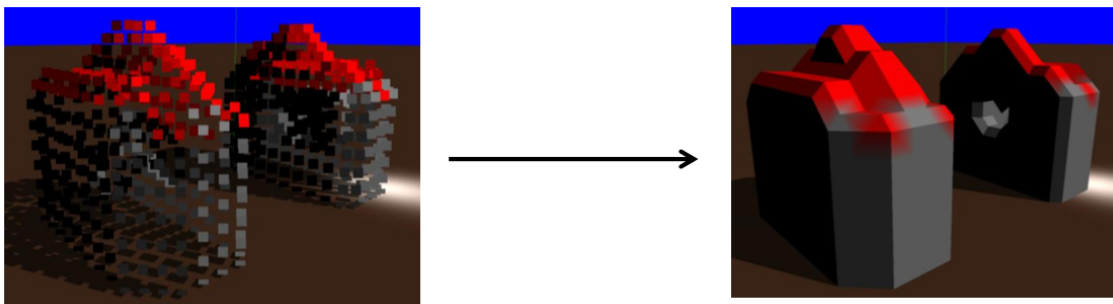


Abbildung 4.7: Die Retriangulierung: Transformation der modifizierten Voxel hin zu einer Dreiecksrepräsentation des beschädigten Gebäudes.

Für die Triangulierung von Voxelmodellen existieren diverse Ansätze und Algorithmen in der einschlägigen Literatur. Ein Anwendungsgebiet ist die Erzeugung von 3D-Modellen (dreiecksbasiert) auf der Basis von Schnittmodellen in der Medizin, beispielsweise bei der Visualisierung bildgebender medizinischer Diagnosegeräte (Beispielsweise Schichtbilder der *Magnet Resonanz Tomographie (MRT)*).

Da es sich bei den zu triangulierenden Voxeln um eine regelmäßige Struktur handelt, bietet eine Variation des *Marching Cube* Triangulierungsalgorithmus die folgenden Vorteile. In der Variante des ursprünglich von Lorensen und Cline [49] vorgestellten Algorithmus zeigen Montani et al. [57], dass eine $2 \times 2 \times 2$ große Voxelregion aus 8 Voxeln auf endlich viele Dreiecksgruppen zurückgeführt werden kann. Diese endlichen Triangulierungsmöglichkeiten sind in Abbildung 4.8 abgebildet. Durch die vorherige Modifikation sind die Positionen bekannt, sodass es sich nicht um die Triangulierung von unbekanntem Punkten handelt, sondern um die Triangulierung von bekannten und regelmäßigen Voxeln. Jede Voxelregion aus 8 Voxeln kann separat betrachtet werden wodurch der Retriangulierungsprozess parallelisierbar wird. Aus diesem Grund wird der Ansatz des *MarchingCube* favorisiert. Der mögliche Nachteil, dass durch die Triangulierung eine unnötig große Anzahl an planaren Dreiecken entsteht, wird in Kapitel 6 diskutiert.

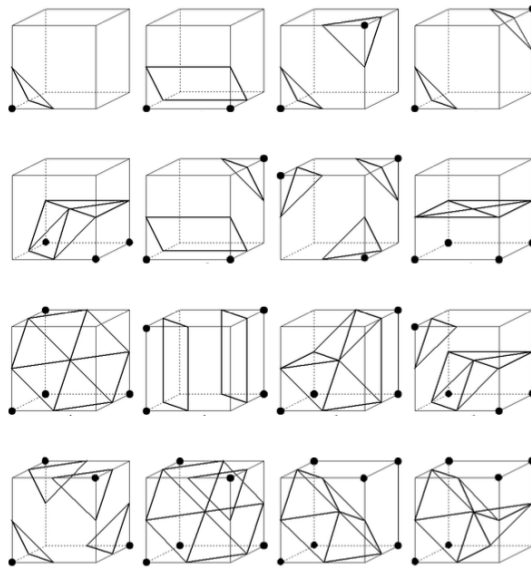


Abbildung 4.8: Endliche Möglichkeiten der Triangulierung einer $2 \times 2 \times 2$ Voxelregion aus Montani et al. [57] (verändert).

Kapitel 5

Implementierungen zum neuen Verfahren

Im vorherigen Kapitel wurden der neue Ansatz zur Handhabung von Gebäudeschäden als Prozess vorgestellt. In diesem Kapitel werden verschiedene Implementierungen zu den Bestandteilen der Prozesskette beschrieben. Mit den Ergebnissen wird verifiziert, dass der zuvor vorgestellte neue Ansatz tatsächlich zur Lösung der Problemstellung genügt und die Handhabung der Gebäudeschäden in der verteilten, virtuellen Simulation verbessert.

5.1 Umsetzung des Verfahrens als 3D-Impactservice

Die Prozesskette vom unbeschädigten zum beschädigten Gebäude (vgl. Abbildung 4.4) ist bisher nur in der Theorie beschrieben worden. Als Proof of Concept werden in diesem Abschnitt Implementierungen zu einzelnen Bestandteilen der Prozesskette vorgestellt, um die Möglichkeiten und Grenzen der Handhabung von Gebäudeschäden mit dem zuvor aufgezeigten Ansatz zu verifizieren. Zu den Komponenten der *Voxelisierung*, der *Berechnung der Schadenstransformation*, der *Anwendung der Schadenstransformation* und der *Retriangulierung* (vgl. Abbildung 4.4) wird jeweils eine prototypische Implementierung als Programmroutine vorgestellt und diskutiert. Ferner werden die Routinen in einem Service, der im Weiteren *3D-Impactservice* genannt wird, für verteilte Simulationen im Sinne des MSaaS Konzeptes (vgl. Abschnitt 2.4.6) zusammengeschaltet.

5.1.1 Implementierungen zur Voxelisierung

Für die Voxelisierung wird ein *grafisches* Verfahren zur Transformation des triangulierten unbeschädigten Gebäudes in Voxel verwendet. *Grafisch* bedeutet, dass das 3D-Gebäudemodell mit einer 3D-Engine zur Anzeige gebracht wird. Ein Grund für ein grafisches Voxelisierungsverfahren ist der im Abschnitt 4.3.1 dargestellte Ansatz, dass die Voxelisierung per Iteration über den Raum im Gegensatz zur Iteration über die Dreiecke des unbeschädigten Gebäudes zu favorisieren ist. Die grafische Voxelisierung entspricht der Iteration über den Raum.

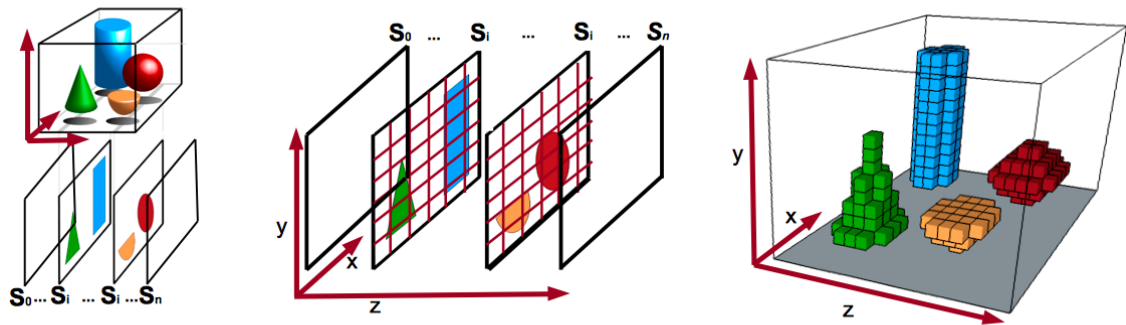


Abbildung 5.1: Voxelisierung per Schnittbildverfahren (Schnittbilder S_0 bis S_n). Linke Abbildung aus [95] (verändert).

Implementierung Die grafische Voxelisierung wird in einer Java Routine implementiert. Vorbild der Implementierung ist das sogenannte *Schnittbildverfahren*. „Ein Schnittbild gibt die inneren Strukturen so wieder, wie sie nach dem Aufschneiden des Objekts oder nach dem Herausschneiden einer dünnen Scheibe vorlägen. Man spricht hier von einer überlagerungsfreien Darstellung der entsprechenden Objektschicht [..]“ [95].

Bei der Voxelisierung eines 3D-Gebäudemodells entspricht die Anzahl der Schnittbilder der Anzahl der Voxel in z-Richtung (zur Achsenbezeichnung vgl. Abbildung 5.1). Jedes Schnittbild ist eine Visualisierung des Gebäudes bei konstanter Tiefe (z-Wert). Die Pixel eines einzelnen Schnittbildes liegen in der x-y-Ebene. Aus allen Schnittbildern wird die Voxelrepräsentation des Gebäudes abgeleitet. Dies ist in Abbildung 5.1 dargestellt.

In bestehenden Schnittbildverfahren, beispielsweise in der Medizin, müssen diese Aufnahmen nacheinander, also schichtweise erzeugt werden. Das 3D-Gebäudemodell würde in der z-Achse iterativ abfotografiert. Je höher die Auflösung der Voxelisierung wäre, desto mehr Schnittbilder müssten erstellt werden und desto mehr Iterationsschritte würden benötigt. In der hier verwendeten Implementierung werden alle Schnittbilder gleichzeitig, also in einem Iterationsschritt, aufgenommen. Für jedes Schnittbild, also für jedes S_i , wird eine 3D-Sichtperspektive, auch Viewpoint genannt, definiert. Die Viewpoints werden so angeordnet, dass sie gleichzeitig dargestellt werden können. Es entsteht eine Multi-Schnittbild, in dem alle Schnittbilder entlang der z-Achse parallel in nur einem Bild dargestellt werden. Ein Multi-Schnittbild enthält alle Schnittbilder S_i . In Abbildung 5.2 sind drei Multi-Schnittbilder verschiedener Auflösungen dargestellt. Für die Multi-Schnittbilder in Abbildung 5.2 wurde das 3D-Gebäudemodell (Haus mit Satteldach und Schornstein) aus Abbildung 4.5 links verwendet.

Im Detail wird die grafische Voxelisierung in der Java basierten 3D-Engine *JMonkeyEngine* implementiert. Zur Voxelisierung wird ein Screenshot des Multi-Schnittbildes gespeichert und in die Voxel transformiert. Dieser Vorgang ist im Programmcode 5.1 auf Seite 77 dargestellt. Die Funktion *Multischnittbild2Voxel*, die ab Zeile 5 definiert ist, transformiert das

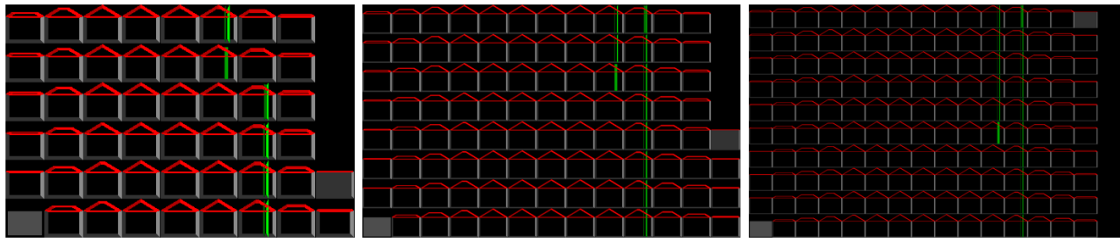


Abbildung 5.2: Darstellung aller Schnittbilder in einem Bild, dem Multi-Schnittbild. Links: 50 Schnittbilder mit jeweils 50x50 Pixeln. Mitte: 100 Schnittbilder mit jeweils 100x100 Pixeln. Rechts: 150 Schnittbilder mit jeweils 150x150 Pixeln. Sortierung der Schnittbilder spaltenweise von unten links nach oben rechts (3D-Gebäudemodell aus Abbildung 4.5 links).

übergebene zweidimensionale Array des Multi-Schnittbildes (Variable *bild*) in das dreidimensionale Array der Voxel (Variable *voxel*).

5.1.2 Implementierungen zur Berechnung der Schadenstransformation

In den Abschnitten 4.2.1 und 4.2.2 wurde dargestellt, dass sowohl die *Parameter des Schadensereignisses* als auch die *Berechnung der Schadenstransformation* stets von den teilnehmenden Simulationssystemen abhängig sind. Die nachfolgende Implementierung einer *Berechnung der Schadenstransformation* ist daher exemplarisch zu verstehen und hat den Zweck die Anwendbarkeit des neuen Ansatzes prototypisch in einem Testszenario zu verifizieren.

Der Implementierung einer *Berechnung der Schadenstransformation* geht voraus, dass *Parameter des Schadensereignisses* bekannt sind. Für die Implementierung eines Testszenarios müssen *Parameter des Schadensereignisses* angenommen werden. Um das Testszenario so realistisch wie möglich zu halten, werden nur Parameter angenommen, die in den bisherigen Vorgehensweisen zur Handhabung von Gebäudeschäden ebenfalls verwendet wurden. Die bisherigen Vorgehensweisen sind das LoS-Konzept (vgl. Abschnitt 1.1) und die Verwendung geometrischer Primitive (beispielsweise Einschlagskrater vgl. Abschnitt 2.4.9).

Es ist offensichtlich, dass für die Verwendung des LoS-Konzept ein Parameter vorhanden sein muss, der die Zuordnung in die Zustandsstufen beschreibt¹. Ferner ist davon auszugehen, dass für die Verwendung von Einschlagskratern ein Parameter der Einschlagsposition bekannt ist. Da beide Parameter (Zustandsstufen und Einschlagsposition) bereits in verteilten, virtuellen Simulationen verwendet wurden, werden sie für das Testszenario herangezogen. Für die *Parameter des Schadensereignisses* wird angenommen, dass die Parameter, Zustandsstufe und Einschlagsposition, bekannt sind.

Außerdem ist davon auszugehen, dass die Position an der das 3D-Gebäudemodell auf dem 3D-Geländemodell platziert wird, die Gebäudeposition, vorhanden ist. Für den Parameter der Zustandsstufe wird, um die Vielfältigkeit vom kleinen Gebäudeschaden bis zur vollständigen Zerstörung des Gebäudes demonstrieren zu können, eine der folgenden Scha-

¹Auch wenn der Parameter nicht direkt von einem Simulationssystem publiziert wird, kann er über Services bereit gestellt werden (vgl. Abschnitt 2.7.3).

Programmcode 5.1 JAVA Quellcodeausschnitt zur Erzeugung der Voxel aus einem Multi-Schnittbild.

```

1  /**@param bild Das Multischnittbild aus den Viewpoints der 3D-Engine
2  * @param aufloesung Die Anzahl der Voxel pro Kante.
3  * @param iVerticalViewpoints Die Anzahl der Schichtbilder pro Spalte.
4  * @return farbige Voxel*/
5  public Color[][][] Multischnittbild2Voxel(Color bild[][], int aufloesung, int iVerticalViewpoints){
6      //Initialisierung der Voxel ;
7      Color [][][] voxel = new Color[aufloesung][aufloesung][aufloesung];
8      //Indizes der Voxel;
9      int x_voxel = 0;
10     int y_voxel = 0;
11     int z_voxel = 0;
12     //Schleifen über alle Pixel des Multischichtbildes;
13     for(int x_pixel = 0; x_pixel < bild.length; x_pixel++) {
14         for(int y_pixel = 0; y_pixel < bild[0].length ;y_pixel++) {
15             if(bild[x_pixel][y_pixel]!=null){
16                 x_voxel = x_pixel%aufloesung;//Modulooperator;
17                 y_voxel = y_pixel%aufloesung;//Modulooperator;
18                 z_voxel = y_pixel/aufloesung + x_pixel/aufloesung * iVerticalViewpoints;
19                 //Uebertragung der Farbwerte des Pixels auf das Voxel;
20                 if(x_voxel<aufloesung && y_voxel<aufloesung && z_voxel<aufloesung){
21                     voxel[x_voxel][y_voxel][z_voxel] = bild[x_pixel][y_pixel];
22                 }
23             }
24         }
25     }
26     return voxel;
27 }

```

den Klassen angenommen²:

Schadensklasse 1: Kein Gebäudeschaden.

Schadensklasse 2: Eine geringer Gebäudeschaden, durch den einige Voxel in geringer Distanz zur Einschlagsposition *entfernt* werden.

Schadensklasse 3: Eine mittlerer Gebäudeschaden (Beispielsweise ein Einsturz des Daches), der sich jedoch nur in einer *Verschiebungsmodifikation* der Voxel äußert.

Schadensklasse 4: Eine schwerer Gebäudeschaden bei dem nur eine Ruine zurückbleibt und ein Großteil des Gebäudes *entfernt* wird.

Die Rechenvorschrift des Testszenarios zur *Berechnung der Schadenstransformation* wird in einer Java Routine implementiert, die im Programmcode 5.2 auf Seite 80 abgebildet ist. Aus der Position des Einschlags wird ein Modifikationsvektorfeld erzeugt, womit die Parameter der Schadenstransformation definiert werden. Als Modifikationsarten werden analog zu den Herleitungen im vorherigen Kapitel (vgl. Abschnitt 4.2.3.2) das *Verschieben* und das *Entfernen* verwendet. Dies wird auch in Abschnitt 5.1.4 noch erläutert.

²Diese Aufteilung wurde in Anlehnung an die im VIntEL Projekt verwendeten Zustandsstufen verwendet (vgl. Abschnitt 2.7.2).

In Programmcode 5.2 wird in Zeile 7 die relative Einschlagsposition aus der Position des 3D-Gebäudemodells abgeleitet. In Abhängigkeit zur übergebenen Schadensklasse wird ab Zeile 13 über die Werte der Variablen *modEntfernenRadius* und *modVerschiebenRadius* festgelegt, bis zu welcher Entfernung zur Einschlagsposition eine Modifikation durchgeführt wird. Die etwaigen Modifikationen werden dann ab Zeile 36 im Modifikationsvektorfeld definiert.

Bewertung Die Rechenvorschrift in Programmcode 5.2 auf Seite 80 stellt ein vereinfachtes Testszenario dar. Die Variabilität der damit erzeugbaren Gebäudeschäden ist, durch die simple Verwendung von vier Schadensklassen, einschränkt. Beim LoS-Konzept werden ebenfalls vordefinierte Schadensklassen verwendet. Die Rechenvorschrift in Programmcode 5.2 zeigt allerdings das Prinzip, wie ein Gebäudeschaden aus unterschiedlichen Modifikationsarten erzeugt werden kann und wie *Parameter des Schadensereignisses* in die Erzeugung des Gebäudeschadens mit einbezogen werden können. Somit stellt bereits das vereinfachte Berechnungsmodell in Programmcode 5.2 eine Verbesserung der Handhabung von Gebäudeschäden dar. Aus diesem Grund wird für die generelle Verifikation des neuen Ansatzes das vereinfachte Berechnungsmodell in Programmcode 5.2 in den nachfolgenden Abschnitten verwendet.

In Anwendungen und Ansätzen, die über die vorliegende Arbeit hinaus gehen, kann dieses vereinfachte Berechnungsmodell erweitert werden. Bei konkreten Simulationen und bekannten Simulationssystemen könnten, bei entsprechender Kenntnis über Munitionstypen, Ballistik oder Waffensysteme, zusätzliche Parameter für die Erzeugung der Modifikationsvektorfelder einbezogen werden. Beispielsweise könnten Parameter wie *Munition Type*, *Final Velocity Vector* oder *Firing Object Identifier*, die im Standard des RPR FOM genannt sind, ausgewertet werden.

5.1.3 Parameter der Schadenstransformation mittels Standards

In Abschnitt 4.2.3.3 wurde gezeigt, dass die Parameter der Schadenstransformation in dem neuen Ansatz einem Modifikationsvektorfeld entsprechen. Wie die Modifikationsvektorfelder definiert werden, wurde noch nicht beschrieben. Im Folgenden wird exemplarisch dargestellt, wie Geodatenstandards zur Definition von Modifikationsvektorfeldern verwendet werden können.

Wie von Averdung [6], Rossmann [65], Stüber [81, 82] und Krückhans [45] gezeigt wurde, können mit der Verwendung von Geodatenstandards in der verteilten Simulationen allgemein erhebliche Mehrwerte erzielt werden. Unter anderem werden in den genannten Quellen die Steigerung der Interoperabilität durch Verwendung des MSaaS-Konzepts, Verbesserung des Fair-Fight durch zentrale Datenbanken und die Transparenz durch offene Standards dargestellt.

Für die Definition von Modifikationsvektorfeldern als Parameter der Schadenstransformation sind diese Eigenschaften von besonderer Bedeutung, da sie den Zielen dieser Arbeit

(Transparenz und bestehende Standards) entsprechen (vgl. Tabelle 3.2). Konkret wird im Ziel der Transparenz in Abschnitt 3.2.4 die Bereitstellung einer Schnittstelle zur Definition von Gebäudeschäden gefordert.

Ein Beispiel einer solchen Schnittstelle zur Definition von Modifikationsvektorfeldern zeigt Programmcode 5.3 auf Seite 81. Darin wird per XML-Code die Definition eines Modifikationsvektors auf Basis der Standards GML [28], Filter Encoding [20] und WFS [94] dargestellt. Im Codeausschnitt wird ein einzelner Vektor eines Modifikationsvektorfeldes mit einer Verschiebungsmodifikation definiert. Der Codeausschnitt besteht aus zwei Blöcken. Der erste Block zwischen den Zeilen 9 und 18 definiert die Verschiebungsmodifikation. Der zweite Block definiert zwischen den Zeilen 19 und 27 das Voxel.

Für die Definition des Voxels wird der Standard des OGC Filter Encoding verwendet. Der Filter definiert einen geometrischen Ausschnitt. Der geometrische Ausschnitt (*BBox*) wird per diagonal entgegengesetzten Punkten (*gml:lowerCorner* und *gml:upperCorner*) beschrieben. Plakativ gesprochen kann der zweite Block wie folgt interpretiert werden: *Selektiere alles innerhalb des Voxels mit dem Minimum [0;0;0] und dem Maximum [1;1;1]*.

Im ersten Block zwischen den Zeilen 9 und 18 wird definiert, wie der Inhalt der Selektion des zweiten Blocks modifiziert werden soll. Per *gml:Vector* im Koordinatensystem EPSG:9606 (Translation) wird eine Verschiebungsmodifikation beschrieben. Plakative Interpretation: *„Verschiebe um den Vektor [0,3;0,3;0,3]“*.

Durch das umschließende Tag *wfs:property* werden die beiden Blöcke kombiniert was zu der Interpretation *„Selektiere alles innerhalb des Voxels mit dem Minimum [0;0;0] und dem Maximum [1;1;1] und verschiebe es um den Vektor [0,3;0,3,0,3]“* führt. Weitere Modifikationen sind in diesem Codeausschnitt nur durch die Zeilen 29 bis 32 angedeutet.

Mit dieser Schnittstelle zur Definition von Parametern der Schadenstransformation, also zur Definition von Modifikationsvektorfeldern, auf der Basis von Geodatenstandards wird der Forderung einer API nachgekommen. Mit der vorgestellten Formalisierung kann jedes Voxel verschoben (*wfs:update*) oder entfernt (*wfs:delete*) werden. Eine im XML Dokument nicht explizit genanntes Voxel, wird entsprechend nicht modifiziert. Auch die Auflösung des Vektorfeldes wird indirekt über die Größe der Voxel definiert.

Zusammenfassend erlaubt die vorgeschlagene Schnittstelle die Definition von beliebigen Modifikationsvektorfeldern, also von beliebigen Parametern der Schadenstransformation. Diese Möglichkeiten können als eine Schnittstelle zur Steuerung der Handhabung von Gebäudeschäden gelten und entsprechen den Anforderungen einer API.

Programmcode 5.2 Java Quellcodeausschnitt einer exemplarischen Berechnung der Schadenstransformation.

```

1 // @param ereignis aus der verteilten Simulation;
2 // @param aufloesung Benoetigte Aufloesung des ModVekFeldes. Anzahl Voxel pro Kante;
3 public Vektorfeld[][][] berechneSchadenstransformation(Schadensereignis ereignis, int aufloesung){
4     //Initialisierung Modifikationsvektorfeld -> Ein Vektor (Point3f) für jedes Voxel;
5     Vektorfeld [][][] vektorFeld = new Vektorfeld[aufloesung][aufloesung][aufloesung];
6     //Relative Einschlagsposition ableiten;
7     Vektor einschlagsPosition = ereignis.einschlagsposition.minus(ereignis.gebäudeposition);
8     //Radius innerhalb dessen eine Entfernenmodifikation vorgenommen wird;
9     float modEntfernenRadius = 0;
10    //Radius innerhalb dessen eine Verschiebungsmodifikation vorgenommen wird;
11    float modVerschiebenRadius = 0;
12    //Vordefinierte Schadensklassen;
13    switch (ereignis.schadensklasse){
14        case "Schadensklasse 1":{
15            modEntfernenRadius = 0;
16            modVerschiebenRadius = 0;
17            break; }
18        case "Schadensklasse 2":{ //Entferne alles im 2m Radius;
19            modEntfernenRadius = 2;
20            modVerschiebenRadius = 0;
21            break; }
22        case "Schadensklasse 3":{//Verschiebe alles im 4m Radius;
23            modEntfernenRadius = 0;
24            modVerschiebenRadius = 4;
25            break; }
26        case "Schadensklasse 4":{//Entferne alles im 10m Radius;
27            modEntfernenRadius = 10;
28            modVerschiebenRadius = 0;
29            break; }
30    }
31    //Erzeugung eines Platzhalters als Entfernenmodifikation;
32    float unendlich = Float.MAX_VALUE;
33    Vektor vekEntfernen = new Vektor(unendlich,unendlich,unendlich);
34    //Erzeugung der Modifikationsvektoren in Abhängigkeit zur Entfernung;
35    //Schleifen über das Modifikationsvektorfeldes;
36    for (int x=0;x<aufloesung;x++){
37        for (int y=0;y<aufloesung;y++){
38            for (int z=0;z<aufloesung;z++){
39                float distanz = getDistance(new Vektor(x, y, z),einschlagsPosition);
40                //Entfernenmodifikation;
41                if(distanz < modEntfernenRadius){
42                    //Maximale Verschiebung = Entfernen;
43                    vektorFeld[x][y][z] = vekEntfernen;
44                }
45                //Verschiebungsmodifikation;
46                else if(distanz < modVerschiebenRadius){
47                    //Verschieben nur entlang y-Achse
48                    //entsprechend der Entfernung;
49                    float verschieb_y = modVerschiebenRadius-distanz;
50                    vektorFeld[x][y][z] = new Vektor(0, -verschieb_y, 0);
51                }
52                else{//Keine Modifikation;
53                    vektorFeld[x][y][z] = new Point3f(0, 0, 0);
54                }
55            }
56        }
57    }
58    return vektorFeld;
59 }

```

Programmcode 5.3 XML Quellcodeausschnitt eines per ISO/OGC Standards definierten Modifikationsvektorfeldes.

```

1 <?xml version="1.0"?>
2 <wfs:Transaction version="1.0.0" service="WFS"
3 xmlns:gml="http://www.opengis.net/gml"
4 xmlns:ogc="http://www.opengis.net/ogc"
5 xmlns:wfs="http://www.opengis.net/wfs"
6 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
7 <!-- FIRST VOXEL MODIFICATION -->
8 <wfs:Update name="Building Damage">
9 <wfs:Property>
10 <wfs>Name>Geometry</wfs>Name>
11 <wfs:Value>
12 <gml:Vector srsDimension="3"
13 srsName="urn:ogc:def:crs:EPSG::9606"
14 axisLabels="X-translation Y-translation Z-translation">
15 <gml:pos>0.3 0.3 0.3</gml:pos>
16 </gml:Vector>
17 </wfs:Value>
18 </wfs:Property>
19 <ogc:Filter>
20 <ogc:BBOX>
21 <ogc:PropertyName>Geometry</ogc:PropertyName>
22 <gml:Envelope srsDimension="3">
23 <gml:lowerCorner>0 0 0</gml:lowerCorner>
24 <gml:upperCorner>1 1 1</gml:upperCorner>
25 </gml:Envelope>
26 </ogc:BBOX>
27 </ogc:Filter>
28 </wfs:Update>
29 <!-- SECOND VOXEL MODIFICATION -->
30 ...
31 <!-- VOXEL n MODIFICATION -->
32 ...
33 </wfs:Update>

```

5.1.4 Implementierungen zur Anwendung der Schadenstransformation

In diesem Abschnitt wird dargestellt, wie die Anwendung der Schadenstransformation implementiert wird. Bei der Anwendung der Schadenstransformation werden die Voxel, die bei der Voxelisierung des unbeschädigten Gebäudes entstehen, durch die entsprechenden Modifikationsvektoren, die den Parametern der Schadenstransformation entsprechen, modifiziert.

In Abschnitt 4.2.3.4 wurde dargestellt, dass für diese Arbeit ein vollbesetztes Modifikationsvektorfeld angenommen wird. Die Verwendung räumlicher Datenstrukturen wie Octrees³ hat unter diesen Voraussetzungen keinen Vorteil. Ein Octree, in dem jedes Blatt einen Modifikationsvektor repräsentieren würde, hätte ebenso viele Blätter wie Datenelemente

³„Ein Octree [...] ist eine Datenstruktur der Informatik. Ein Octree ist ein gewurzelter Baum, dessen Knoten jeweils entweder acht direkte Nachfolger oder gar keine Nachfolger haben. Octrees werden hauptsächlich in der Computergrafik verwendet, um dreidimensionale Datensätze hierarchisch zu untergliedern“ [95].

eines vergleichbaren Arrays. Der Aufbau eines Octrees benötigt jedoch mehr Laufzeit als die Speicherreservierung eines Array. Aus diesen Gründen wurden sowohl bei der Voxelisierung (vgl. Programmcode 5.1) als auch für die Modifikationsvektorfelder, die bei der Berechnung der Schadenstransformation definiert werden (vgl. Programmcode 5.2), Arrays als Datenstruktur verwendet.

Implementierung Die Implementierung zur Anwendung der Schadenstransformation ist unter der Annahme, dass Arrays verwendet werden, trivial. Gegeben sind die beiden Arraydatenstrukturen $Voxel_{xyz}$ und $Vektorfeld_{xyz}$. Jedem $Voxel_{xyz}$ kann exakt ein Modifikationsvektor $Vektor_{xyz}$ aus dem Modifikationsvektorfeld $Vektorfeld_{xyz}$ zugeordnet werden. Die Modifikation wird innerhalb einer Java Routine durch ein drittes Array, dem sogenannten *Schadensarray*, implementiert. Innerhalb der Routine wird über die $Voxel_{xyz}$ iteriert. Für jedes $Voxel_{xyz}$ wird untersucht, welche Modifikation $Vektor_{xyz}$ an der entsprechenden Position im Modifikationsvektorfeld $Vektorfeld_{xyz}$ vorliegt. Für jedes $Voxel_{xyz}$ ergeben sich die folgenden Möglichkeiten, deren Resultate im Schadensarray gespeichert werden:

1. **Keine Modifikation:** $Vektor_{xyz}$ des Modifikationsvektorfeldes hat die Länge 0
 \Rightarrow Kopieren des Voxels durch die Zuordnung $Schadensarray_{xyz} = Voxel_{xyz}$.
2. **Entfernen-Modifikation:** $Vektor_{xyz}$ des Modifikationsvektorfeldes ist *unendlich*
 \Rightarrow Entfernen des Voxels durch die Zuordnung $Schadensarray_{xyz} = null$.
3. **Verschieben-Modifikation:** Die Länge von $Vektor_{xyz}$ des Modifikationsvektorfeldes ist größer als 0 und kleiner als *unendlich*
 \Rightarrow Verschieben durch die Zuordnung $Schadensarray_{xyz} = Voxel_{xyz} + Vektor_{xyz}$.

5.1.5 Implementierungen zur Retriangulierung

Das zuvor beschriebene Schadensarray beinhaltet die Daten für den Schritt der Retriangulierung. Für die Retriangulierung wird, wie bereits im Abschnitt 4.3.3 begründet wurde, der Marching Cube Algorithmus verwendet und implementiert.

Während der Triangulierung wird die Datenstruktur des Schadensarray iterativ durchlaufen. Jede 2x2x2 Voxelregion wird separat betrachtet. Ein einzelner Voxel des Schadensarray ist entweder leer oder belegt. Für die geometrische Retriangulierung ist nur die binäre Information *leer* oder *nicht leer* von Interesse. Jede betrachtete Voxelregion besteht aus 8 Voxeln, die jeweils gefüllt oder leer sein können. Daraus ergibt sich für jede Voxelregion eine endliche Anzahl an Konstellationen, nämlich 2^8 also 256 Möglichkeiten. Diese 256 verschiedenen Konstellationen können, wie von Montani et al. [57] gezeigt, 16 Hauptgruppen zugeordnet werden (vgl. Abbildung 4.8). In der hier implementierten Umsetzung werden die Dreiecke aller möglichen Konstellationen statisch vorgehalten. Während der Retriangulierung erfolgt pro Voxelregion die Bestimmung der zugeordneten Konstellation. Da die statischen Dreiecke der Konstellationen am Ursprung definiert sind, werden alle Punkte

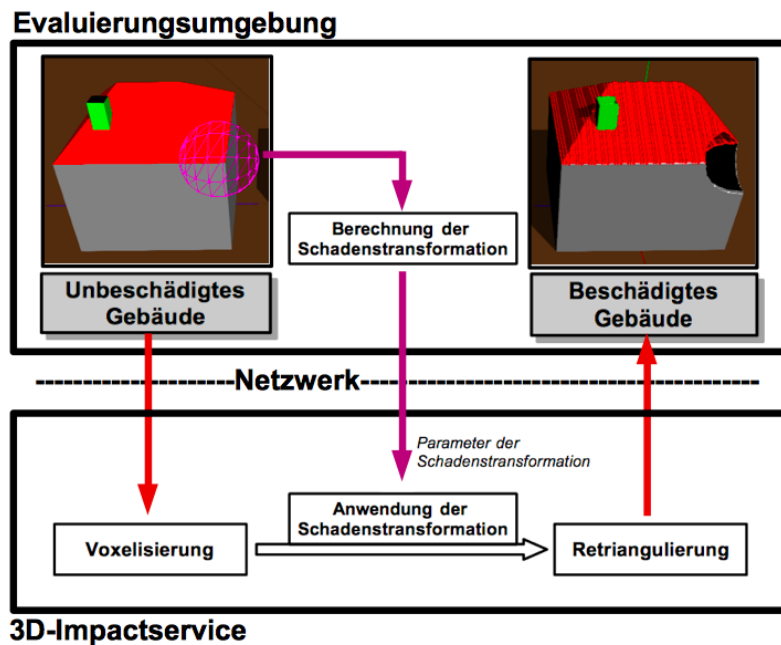


Abbildung 5.3: Evaluierungsumgebung zum Test des 3D-Impactservice.

der Dreiecke kopiert und durch eine Translation um die Position der aktuellen Voxelregion verschoben. Das Ergebnis der Retriangulierung ist ein Vektor aus Dreiecken. Dies ist die Geometrie des beschädigten Gebäudes. Da die Anzahl der Dreiecke von der Konstellation abhängig ist (vgl. Abbildung 4.8) und variieren kann, wird hier die Datenstruktur eines Vektors (einer Liste) bevorzugt.

Die Farben der Dreiecke werden aus den Farbwerten der Voxel verwendet und bei Überbestimmung für die Punkte der Dreiecke interpoliert.

5.1.6 Aggregation der Routinen zu einem 3D-Impactservice

Die beschriebenen Routinen der Voxelisierung, der Modifikation (Anwendung der Schadenstransformation) und der Retriangulierung werden in einer Java Routine zusammengefasst. Dies ermöglicht die Interpretation dieser Routine als Service in einer verteilten Umgebung konform zum M&SaaS-Konzept (vgl. Abschnitt 2.4.6). Der Service wird im Folgenden als *3D-Impactservice* bezeichnet und ist in Abbildung 5.3 unten dargestellt.

Der 3D-Impactservice hat zwei Input- und ein Outputparameter. Dem Service werden ein 3D-Gebäudemodell (das unbeschädigte 3D-Gebäudemodell) und das Modifikationsvektorfeld (Parameter der Schadenstransformation) übergeben. Anschließend erfolgt die Abarbeitung der Prozesskette *Voxelisierung*, *Modifikation* und *Retriangulierung* seitens des Services. Final wird das modifizierte und retriangulierte 3D-Gebäudemodell, das beschädigte Gebäude, vom Service zurückgegeben.

5.2 Verteilte Evaluierungsumgebung

In diesem Abschnitt werden die Implementierung und die Ergebnisse einer Test- und Evaluierungsumgebung für den 3D-Impactservice vorgestellt. Mit der Test- und Evaluierungsumgebung wird der neue Ansatz zur Handhabung von Gebäudeschäden verifiziert. Bei den Untersuchungen steht, neben der Authentizität des Gebäudeschadens (vgl. Abschnitt 3.2.1), die Anforderung der Reaktionszeit (vgl. Abschnitt 3.2.3) im Vordergrund.

5.2.1 Beschreibung der Evaluierungsumgebung

Der zuvor aufgezeigte 3D-Impactservice wird zur Verifikation des neuen Ansatzes verwendet. Dazu wird eine gekapselte Evaluierungsumgebung implementiert und mit dem 3D-Impactservice gekoppelt. Beide Komponenten, die Evaluierungsumgebung und der 3D-Impactservice, sind über eine Netzwerkinfrastruktur verbunden und kommunizieren über Netzwerksockets, so dass die Bedingungen einer verteilten Simulation nachgestellt werden. Die Gesamtarchitektur aus 3D-Impactservice und Evaluierungsumgebung ist in Abbildung 5.3 dargestellt.

Die Evaluierungsumgebung ist in Java implementiert. In der Software wird ein typisches 3D-Gebäudemodell bestehend aus Boden-, Wand- und Dachflächen sowie einem Schornstein, erzeugt und visualisiert. Das exemplarische 3D-Gebäudemodell ist beispielsweise in Abbildung 5.3 unten links dargestellt.

Berechnung der Schadenstransformation Mit der Evaluierungsumgebung können interaktiv Schadensereignisse simuliert werden. Die Erzeugung von Modifikationsvektorfeldern erfolgt interaktiv mit einem Auswahlwerkzeug am visualisierten unbeschädigten Gebäude. Bei einem Klick auf das 3D-Gebäudemodell wird an der Klickposition des 3D-Gebäudemodells eine virtuelle Sphäre eingeblendet, die anschließend per Mausradsteuerung im Radius skaliert werden kann. Aus dieser interaktiv wählbaren Sphäre werden die Parameter der Schadenstransformation abgeleitet, in dem beispielsweise ein Modifikationsvektorfeld der Schadensklasse 2 (vgl. Abschnitt 5.1.2) erzeugt wird. Die Schadensklasse 2 beinhaltet das Entfernen von Voxeln. Es wird ein Modifikationsvektorfeld erzeugt, bei dem alle Voxel, die sich innerhalb der Sphäre befinden, entfernt werden. Durch diese Vorgehensweise ist die interaktive Erzeugung von Modifikationsvektorfeldern möglich. Neben der Schadensklasse 2 sind auch die weiteren Schadensklassen der Rechenvorschrift aus Programmcode 5.2 implementiert und wählbar.

Aufruf des 3D-Impactservice Nach der interaktiven Erzeugung eines Modifikationsvektorfeldes wird dieses zusammen mit dem unbeschädigten Gebäude per Netzwerksockets an den 3D-Impactservice übermittelt. Der 3D-Impactservice berechnet auf Basis dieser beiden Input-Parameter (Modifikationsvektorfeld und unbeschädigtes Gebäude) das beschädigte Gebäude per Voxelisierung, Modifikation und Retriangulierung. Anschließend wird das beschädigte Gebäude an die Evaluierungsumgebung zurückgeliefert.

Ergebnis des 3D-Impactservice Das erhaltene beschädigte Gebäude wird in der Evaluierungsumgebung neben dem unbeschädigten Gebäude dargestellt, so dass beide 3D-Gebäudemodelle unmittelbar miteinander verglichen werden können.

In Abbildung 5.3 rechts oben ist zu erkennen, dass im beschädigten 3D-Gebäudemodell die Voxel entsprechend zur Sphäre entfernt wurden. Im beschädigten Gebäude ist eine Öffnung im 3D-Gebäudemodell dargestellt, die in Position und Größe der gewählten Sphäre entspricht.

Die Evaluierungsumgebung ermöglicht zusammengefasst die synthetische Erzeugung von Gebäudeschäden und erlaubt, neben der Untersuchung des Laufzeitverhaltens, den unmittelbaren optischen Vergleich von unbeschädigtem und beschädigtem 3D-Gebäudemodell. Der neue Ansatz zur Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation kann dadurch getestet und verifiziert werden.

5.2.2 Testreihen und Untersuchung des Laufzeitverhaltens

In diesem Abschnitt wird die Durchführung mehrerer Testreihen des 3D-Impactservice mit der Evaluierungsumgebung beschrieben. Das Ziel der Testreihen ist eine Bewertung des 3D-Impactservice und somit des neuen Ansatzes im Hinblick auf die Anforderungen der Authentizität und der Reaktionszeit. In diesem Abschnitt steht die Durchführung der Testreihen im Vordergrund. Die Bewertung der Ergebnisse wird im nachfolgenden Abschnitt beschrieben.

Messung des Laufzeitverhalten Bezüglich des Laufzeitverhaltens ist nicht nur der Prozess des 3D-Impactservice von Interesse. Die Summe der Laufzeiten aller Teilprozesse wird benötigt, um die Erfüllung der Anforderung der Reaktionszeit bewerten zu können. Dazu wird der komplette Prozess der Erzeugung des Gebäudeschadens in 6 Phasen unterteilt:

Phase 1: Erstellen des Modifikationsvektorfeldes aus der Sphäre.

Phase 2: Übertragung von unbeschädigtem Gebäude und Modifikationsvektorfeld.

Phase 3: Voxelisieren innerhalb des 3D-Impactservice.

Phase 4: Modifizieren innerhalb des 3D-Impactservice.

Phase 5: Retriangulieren innerhalb des 3D-Impactservice.

Phase 6: Ausliefern des beschädigten Gebäudes an die Evaluierungsumgebung.

Durchführung der Testreihen Für die Laufzeittest wurden die drei in Abschnitt 5.1.2 genannten Schadensstufen verwendet. Zu jeder Schadensstufe wurden die Laufzeiten der 6 Phasen, bei variierenden Auflösungen der Modifikationsvektorfelder, gemessen. Die Auflösungen wurden zwischen 20^3 und 250^3 Voxeln variiert. 20^3 stellt die minimale Auflösung dar, bei der noch ein optischer Gebäudeschaden visuell feststellbar ist. Die Auflösung von

250^3 stellt eine Obergrenze dar, die noch auf aktuellen Standard PCs berechnet werden kann. Alle Tests wurden auf einem Windows PC mit Intel Core i5-3470 3,2 GHz, 8 GB RAM und dem Grafikprozessor GeForce GT 630 mit 1GB dediziertem Grafikspeicher durchgeführt.

In Summe wurden drei Testreihen durchgeführt. Jede Testreihe verwendet eine der drei Schadensstufen aus Abschnitt 5.1.2. Pro Testreihe wurden die Auflösungen 20, 50, 100, 150, 200 und 250 Voxel verwendet. Für jede Auflösung sind jeweils die Laufzeiten der 6 Phasen gemessen worden.

Für jede Auflösung und jede Schadensstufe werden, neben der Laufzeitmessung der sechs Phasen, sowohl die unbeschädigten als auch die beschädigten Gebäude visualisiert. Diese Visualisierungen sind in Abbildung 5.4 für die Schadensstufe 2, in Abbildung 5.5 für Schadensstufe 3 und in Abbildung 5.6 für Schadensstufe 4 dargestellt. Die Abbildungen zeigen jeweils das unbeschädigte Gebäude mit der selektierten Sphäre zur Erzeugung des Modifikationsvektorfeldes und das beschädigte Gebäude.⁴

In den folgenden Tabellen sind die Laufzeiten der verschiedenen Phasen zu den unterschiedlichen Auflösungen und Schadensmodellen dargestellt. Konkret werden in Tabelle 5.1 die Laufzeiten zu Schadensmodell 2, in Tabelle 5.2 die Laufzeiten zu Schadensmodell 3 und in Tabelle 5.3 die Laufzeiten zu Schadensmodell 4 dargestellt. Alle Zeitangaben in den Tabellen sind in Millisekunden.

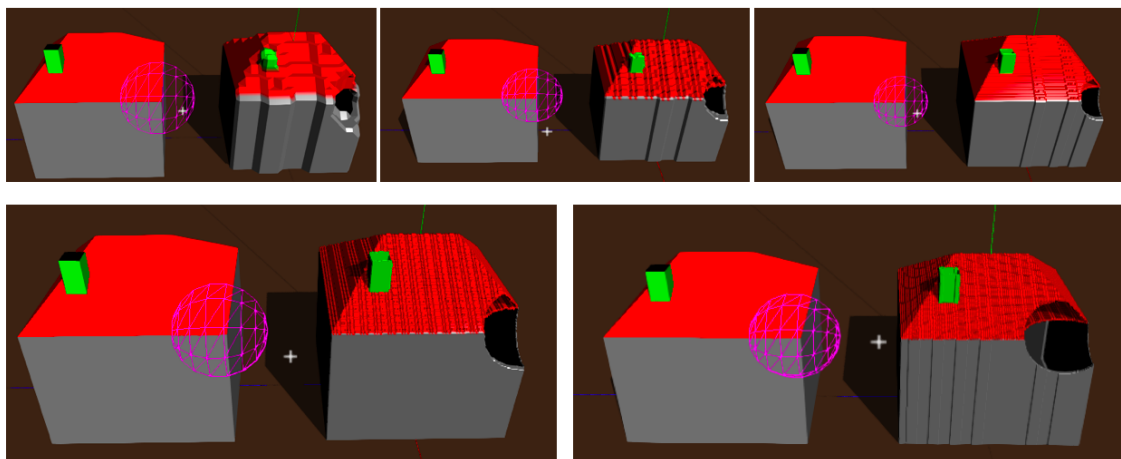


Abbildung 5.4: Vergleichende Darstellung von fünf Gebäudeschäden variierender Auflösung mit Schadensstufe 2 (slight damage). Von links oben nach rechts unten: 20^3 , 50^3 , 100^3 , 150^3 und 200^3 Voxel. Zugehörige Laufzeiten und die Anzahl der Dreiecke sind in Tabelle 5.1 dargestellt.

⁴Im Unterschied zu den tabellarischen Auflistungen der Laufzeiten enthalten die Abbildungen nur Auflösungen bis 200^3 Voxel. Die maximale Auflösung von 250^3 Voxel kann zwar problemlos berechnet werden, die Visualisierung ist jedoch nur bedingt möglich. Um die Unsicherheit von Netzwerklatenzen zu vermeiden, werden 3D-Impactservice und Evaluierungsumgebung auf dem identischen System ausgeführt. Beide Programme teilen sich also eine Grafikhardware. Dies halbiert die zur Verfügung stehenden Speicher und Prozessor-Ressourcen. Die Auflösung von 250^3 Voxel kann unter diesen Bedingungen auf der verwendeten Hardwarekonstellation nicht mehr visualisiert werden. Dieses Problem wird auch im finalen Abschnitt diskutiert (vgl. Abschnitt 6.1).

#Voxel	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Summe	#Dreiecke
20^3	2	1	60	1	2	12	78	0,006 Mio
50^3	3	2	60	1	13	55	134	0,061 Mio
100^3	21	45	90	11	64	188	419	0,24 Mio
150^3	97	123	111	37	178	478	1024	0,6 Mio
200^3	192	227	115	115	502	981	2132	1,2 Mio
250^3	422	416	122	250	900	1893	4003	1,8 Mio

Tabelle 5.1: Vergleich der Laufzeiten in Millisekunden sowie Anzahl der Dreiecke der exemplarischen Schadensstufe 2 (*slight damage*) bei Variation der Auflösung des Vektorfeldes. Zugehörige Visualisierung der Gebäudeschäden in Abbildung 5.4.

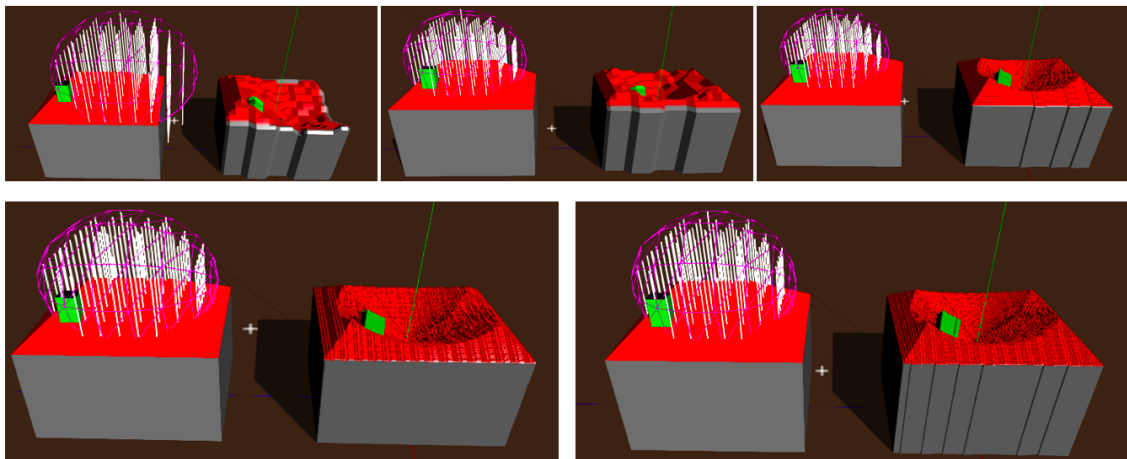


Abbildung 5.5: Vergleichende Darstellung von fünf Gebäudeschäden variierender Auflösung mit Schadensstufe 3 (*medium damage*). Von links oben nach rechts unten: 20^3 , 50^3 , 100^3 , 150^3 und 200^3 Voxel. Zugehörige Laufzeiten und die Anzahl der Dreiecke sind in Tabelle 5.2 dargestellt.

5.2.3 Bewertung der Ergebnisse

Ergebnisse des Laufzeitverhaltens In den abgebildeten Tabellen (5.1, 5.2 und 5.3) steigen die Laufzeiten mit der verwendeten Auflösung. Je mehr Voxel zu berechnen, zu modifizieren und zu retriangulieren sind, desto höher ist der rechnerische Aufwand und die Laufzeit des Prozesses. Ebenfalls steigt die Übertragungszeit zwischen Evaluierungsumgebung und 3D-Impactservice mit der Anzahl der Modifikationsvektoren und der zurückzuliefernden Dreiecke des beschädigten Gebäudes. Je mehr Dreiecke oder Modifikationsvektoren übermittelt werden müssen, desto höher sind die Laufzeiten der Datenübertragung in den Phasen 1 und 6.

Grundsätzlich sind die Laufzeiten der einzelnen Phasen erst ab etwa 100^3 Voxel interpretierbar.⁵ Bei 20^3 und 50^3 kann nur festgestellt werden, dass die Berechnungszeit der Voxelisierung zwar geringer ist als bei größeren Auflösungen, jedoch im Vergleich zu den

⁵Da die Hintergrundaktivitäten des Betriebssystems nicht exakt messbar und somit die Rahmenbedingungen nie exakt reproduzierbar sind, können Messungenauigkeiten nicht ausgeschlossen werden. Dementsprechend sind Ungenauigkeiten zwischen 10 bis 20%, jedoch mindestens 20 Millisekunden, bei der Interpretation zu berücksichtigen.

#Voxel	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Summe	#Dreiecke
20^3	2	3	40	2	1	9	57	0,006 Mio
50^3	1	6	31	1	13	51	103	0,058 Mio
100^3	17	44	40	11	65	208	385	0,25 Mio
150^3	61	102	80	38	189	503	973	0,6 Mio
200^3	173	220	91	115	541	919	2059	1,2 Mio
250^3	368	401	119	233	932	1928	3981	1,9 Mio

Tabelle 5.2: Vergleich der Laufzeiten in Millisekunden sowie Anzahl der Dreiecke der exemplarischen Schadensstufe 3 (*medium damage*) bei Variation der Auflösung des Vektorfeldes. Zugehörige Visualisierung der Gebäudeschäden in Abbildung 5.5.

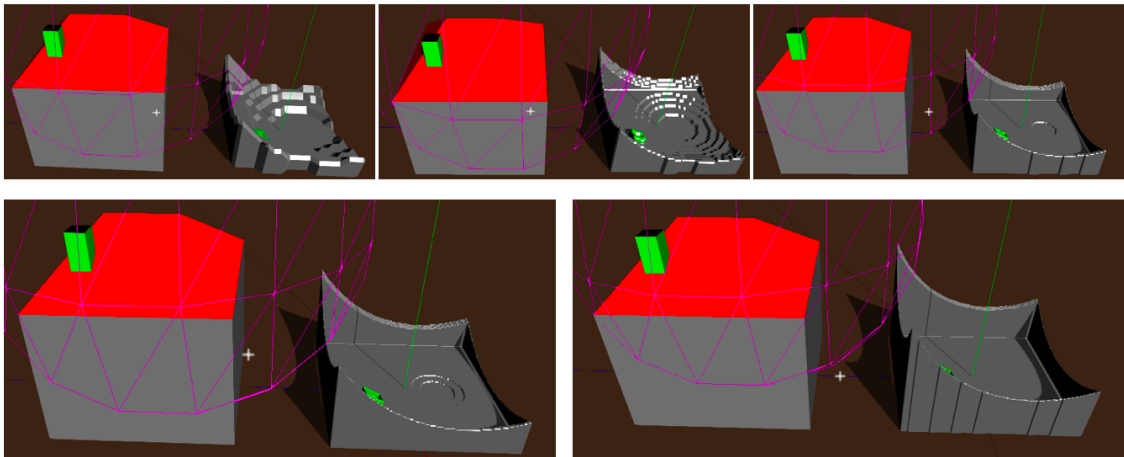


Abbildung 5.6: Vergleichende Darstellung von fünf Gebäudeschäden variierender Auflösung der Schadensstufe 4 (*heavy damage*). Von links oben nach rechts unten: 20^3 , 50^3 , 100^3 , 150^3 und 200^3 Voxel. Zugehörige Laufzeiten und die Anzahl der Dreiecke sind in Tabelle 5.3 dargestellt.

anderen Phasen nicht gegen 0 geht. Dies ist auf die Erzeugung der Multi-Schnittbilder zurückzuführen. Unabhängig von der Auflösung muss der Ausgangszustand mittels 3D-Engine in den Grafikspeicher überführt und zur Anzeige gebracht werden. Dieser Aufwand bleibt auch bei 20^3 und 50^3 Voxeln bestehen und ist auflösungsunabhängig feststellbar.

Da sich die Anzahl der Schnittbilder mit der Anzahl der Voxel erhöht, steigt der Aufwand der Voxelisierung mit zunehmender Auflösung. Diese Beobachtung ist unabhängig von der Art des Schadens, also in allen Tabellen erkennbar.

Bei höheren Auflösungen sind die Laufzeiten der Phasen durchaus unterschiedlich. So sind die Steigerungen der Laufzeiten der Phasen 1, 3 und 4 deutlich geringer als die Laufzeiten der Phasen 5 und 6. Dies ist auf den höheren Speicherbedarf der Dreiecke zurückzuführen. Während ein einzelnes Voxel quasi eine punkthafte Information darstellt und durch 6 Parameter definiert ist (3 Positionen und 3 Farbwerte), werden für die Darstellung des Voxels (vgl. Abbildung 4.8) mindestens 8 Dreiecke verwendet. Der höhere Speicherbedarf und die in der Konsequenz längeren Rechen- und Auslieferungszeiten begründen die Unterschiede in den Phasen.

In Summe ist zu beobachten, dass sich die Laufzeiten in Abhängigkeit der Schadensstufe nur

#Voxel	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Summe	#Dreiecke
20 ³	1	1	30	1	1	8	42	0,002 Mio
50 ³	4	6	20	1	6	16	53	0,017 Mio
100 ³	26	38	50	10	35	66	225	0,079 Mio
150 ³	91	103	70	37	111	134	546	0,172 Mio
200 ³	223	189	110	117	268	314	1221	0,394 Mio
250 ³	565	370	125	222	615	506	2403	0,5 Mio

Tabelle 5.3: Vergleich der Laufzeiten in Millisekunden sowie Anzahl der Dreiecke der exemplarischen Schadensstufe 4 (*heavy damage*) bei Variation der Auflösung des Vektorfeldes. Zugehörige Visualisierung der Gebäudeschäden in Abbildung 5.6.

wenig ändern. Zwischen den drei Tabellen sind nur geringfügige Unterschiede festzustellen. Diese minimalen Unterschiede sind durch die geringere Anzahl der Dreiecke zu erklären. Die ruinenartige Darstellung des Schadensmodells 4 besteht aus weniger Dreiecken als die leichte Beschädigung des Schadensmodells 2. Aus diesem Grund sind sowohl die Laufzeiten der Retriangulierung, als auch die Laufzeiten der Auslieferung bei Schadensstufe 4 geringer.

Bewertung der Ergebnisse des Laufzeitverhaltens Das Laufzeitverhalten der Prozesskette ist vor dem Hintergrund des benötigten Hauptspeichers während der Routinen zu bewerten. Dies wird an zwei Beispielen verdeutlicht:

Beispiel 1: Ein Multi-Schnittbild der Voxelisierung hat bei einer Auflösung von 250³ Voxel eine Größe von 5250x3250 Pixeln. Bei dem der JMonkeyEngine zugrunde liegenden Pixel Format von 4 Byte pro Pixel ergibt sich daraus ein permanenter Speicherbedarf von 65 *Megabyte (MB)*. Dieser Speicherbedarf ist unabhängig von der Geometrie des unbeschädigten Gebäudes. Während der Voxelisierung müssen immer 65 MB gelesen, transformiert und visualisiert werden.

Beispiel 2: Der Endzustand der Schadensstufe 3 wurde mit einer Anzahl von 1,9 Millionen Dreiecken berechnet. Auf der Basis der Notation eines Dreiecks in einer OpenGL basierten 3D-Engine werden für jedes Dreieck 27 Parameter benötigt⁶. Unter der Voraussetzung, dass jeder Parameter im Datentyp *Float* gespeichert wird und dieser einen Speicherbedarf von 4 *Byte* hat, ergibt sich für den Gebäudeschaden ein Speicherbedarfs von $1900000 \cdot 27 \cdot 4 \text{ byte} = 195 \text{ Megabyte}$. Innerhalb der Laufzeit von weniger als 4 Sekunden werden nicht nur Daten in der Größe von 65 MB für die Voxelisierung adressiert, sondern diese auch zu 195 MB des beschädigten Gebäudes transformiert und an die Evaluierungsumgebung ausgeliefert. Da trotz dieser Größe eine Laufzeit von 5 Sekunden nicht überschritten wird, ist festzustellen, dass der Ansatz grundsätzlich, und diese Implementierung im Speziellen, bezüglich der Laufzeit der Anforderung der Reaktionszeit genügt.

Es bleibt fraglich, ob eine Gebäudeschaden tatsächlich mit einer Auflösung von 250³ Voxel berechnet werden sollte oder auch eine reduzierte Auflösung zur Verdeutlichung des Gebäudeschadens ausreicht. Das Problem der Anzahl der Dreiecke wird im Abschnitt 6.1 thematisiert.

⁶3 Punkte pro Dreieck. 9 Parameter pro Punkt: 3 Koordinaten, 3 Farbwerte, 3 Normalenrichtungen.

Bewertung der Authentizität In Abschnitt 3.2.1 wurde definiert, dass ein authentischer Gebäudeschaden auf den **Eigenschaften des Gebäudes (1)** und den **Eigenschaften des Waffeneinsatzes (2)** basiert. Hinsichtlich dieser Anforderungen kann festgestellt werden, dass die Gebäudeschäden in den Testreihen authentisch sind, was nachfolgend begründet wird:

1. Die **Eigenschaften des 3D-Gebäudemodells** werden berücksichtigt, in dem die Geometrie und die Farbe der 3D-Gebäudemodelle bei der Anwendung der Schadenstransformation verwendet wird.
2. Die **Eigenschaften des Waffeneinsatzes** werden berücksichtigt indem die zur Laufzeit der Simulation zur Verfügung stehenden Informationen eines Schadensereignisses, den Gebäudeschaden beeinflussen können, beispielsweise durch Auswertung von Parametern des Schadensereignisses.

Über diese Anforderungen hinaus muss jedoch festgestellt werden, dass die Authentizität des Gebäudeschadens, beispielsweise in den Abbildungen 5.4, 5.5 oder 5.6, hinter der grafischen Qualität der in Abschnitt 2.3.3 aufgezeigten und beispielsweise in Abbildung 2.3 dargestellten 3D-Gebäudemodellen zurückbleibt. Zwar repräsentieren die dargestellten beschädigten Gebäude eindeutig das Schadensereignis, allerdings haben die Abbildungen auf Grund des ebenen Geländes oder des einfarbigen Hintergrundes noch einen prototypischen Charakter. Durch die Absenz von grafischen Effekten erscheint das beschädigte 3D-Gebäudemodell nicht authentisch. Die Authentizität wird im folgenden Abschnitt 5.3 mit der Einbindung des 3D-Impactservice in eine reale Simulationssoftware optimiert.

5.3 Anbindung in eine reale Simulationssoftware

In diesem Abschnitt wird gezeigt, wie die Implementierung des 3D-Impactservice, neben der zuvor dargestellten Einbindung in eine Evaluierungsumgebung, auch in einer echten Simulationssoftware zum Einsatz kommen kann.

5.3.1 Simulationsszenario in X-Plane

Der zuvor beschriebene 3D-Impactservice wird mit der Flugsimulationssoftware X-Plane in einer verteilten Simulation gekoppelt. X-Plane ist eine Flugsimulationssoftware, die sich durch hohen Realismus und Erweiterbarkeit (*Customizing*) auszeichnet [95]. Die Erweiterbarkeit von X-Plane ermöglicht individuelle Veränderungen an der virtuellen Umwelt und an den Luftfahrzeugen über Plugins.

Für die Kopplung von X-Plane und dem 3D-Impactservice wird ein militärisches Flugzeug, das als Standardmodell in X-Plane verfügbar ist, auf einem virtuellen Flughafen in X-Plane positioniert. Das Flugzeugmodell wird virtuell mit Raketen des Typs AGM-65 Maverick ausgerüstet. Während der Simulation werden diese Raketen auf ein 3D-Gebäudemodell gerichtet und ausgelöst. Die Positionen von Flugzeug und Gebäude werden so gewählt, dass

die Raketen in unmittelbarer Nähe zum Gebäude das virtuelle Gelände berühren und detonieren.⁷

5.3.2 Implementierung in X-Plane

Für die Kopplung des 3D-Impactservice wird in X-Plane ein Plugin auf der Basis der *xsquawkbox*⁸ implementiert. Innerhalb des Plugins werden zyklisch der Zustand und die Position aller Raketen abgefragt. Wechselt der Zustand einer Rakete von *in der Luft* zu *detoniert*, wird die letzte Position der Rakete relativ zum Standort des Gebäudes als Parameter eines Schadensereignisses verwendet. Die letzte Position der Rakete (*Detonation Location*) wird als eine Nachricht codiert, die der *Munition Detonation Interaction* des RPR FOM entspricht, und an den 3D-Impactservice gesendet. Innerhalb des 3D-Impactservice werden die codierte Nachricht in ein Modifikationsvektorfeld umgesetzt (Berechnung der Schadenstransformation), sowie die Prozessschritte Voxelisierung, Modifikation und Retriangulierung durchgeführt. Im Detail werden während der Simulation die folgenden Prozessschritten durchlaufen:

1. Vor der Simulation: Positionieren eines X-Plane Flugzeugs in Schussweite des Beispielgebäudes.
2. Start der Simulation: Abschuss einer Rakete im Waffensystem.
3. Abfangen der Detonationsposition (in einem entwickelten X-Plane-Plugin) sowie Erzeugung einer *Munition Detonation Interaction* mit dem Parametern der *Detonation Location*.
4. Übermittlung der *Munition Detonation Interaction* und des unbeschädigten 3D-Gebäudemodells an den 3D-Impactservice.
5. Erzeugung des Modifikationsvektorfeldes aus der *Munition Detonation Interaction* sowie Voxelisierung, Modifikation und Retriangulierung des unbeschädigten Gebäudes hin zu einem beschädigten Gebäude.
6. Auslieferung des beschädigten Gebäudes in einem X-Plane konformen 3D-Format (eine Datei pro Gebäude). Kopieren der Datei in das X-Plane Installationsverzeichnis.
7. Trigger vom 3D-Impactservice an X-Plane zum erneuten Laden des Gebäudes (Austausch des unbeschädigten Gebäudes durch ein beschädigtes Gebäude).
8. Das beschädigte Gebäude wird dargestellt und repräsentiert die Parameter der Schadensereignisses.

⁷Da X-Plane per se nicht den Beschuss von Gebäuden unterstützt und eine Rakete nur auf dem Gelände zur Explosion geführt werden kann, werden Entfernung und Abschusswinkel in etwa so gewählt, dass die Rakete in der Nähe des Gebäudes einschlagen wird. Die Einschlagsposition ist trotzdem zufällig und nicht reproduzierbar. Da X-Plane auch Witterungseinflüsse wie Wind oder andere Wechselwirkungen (Flugzeug wackelt leicht bei Abschuss) berücksichtigt, sind die exakten Einschlagspositionen nie identisch.

⁸www.xsquawkbox.net - letzter Zugriff 07.05.2018

9. Ende der Simulation oder weiterer Beschluss unter Verwendung des beschädigten Gebäudes (weitere Beschädigung des bereits beschädigten Gebäudes, sogenannter *Dirty Battlefield Effect* nach Tolk [88]).

Als Rechenvorschrift zur *Berechnung der Schadenstransformation* wird eine Kombination der in Abschnitt 5.1.2 beschriebenen Schadensstufen 2 und 3 verwendet. Die Rechenvorschrift des 3D-Impactservice wird nach subjektiven Eindrücken mit dem Ziel erstellt, den Gebäudeschaden zu Demonstrationszwecken so authentisch wie möglich erscheinen zu lassen. Das Modifikationsvektorfeld wird sowohl aus Einschlagsposition (*Detonation Location*) und drei statischen Radien mit den Werten $r_1 = 3,5\text{ m}$, $r_2 = 7\text{ m}$ und $r_3 = 7,5\text{ m}$ berechnet:

- *Entfernen* aller Voxel bis zu einem Radius r_1 um die Einschlagsposition.
- *Verschieben* aller Voxel mit einem Radius zwischen r_1 und r_2 auf die Geländehöhe zur Erzeugung von versprengten Bruchstücken (Trümmer).
- *Verschieben* aller Voxel mit einem Radius zwischen r_2 und r_3 um $0,2\text{ m}$ zur Visualisierung eines Ausfransungseffektes am 3D-Gebäudemodell.
- Keine Modifikation aller Voxel mit einem Radius größer r_3 .

5.3.3 Bewertung der Ergebnisse

Die in X-Plane erzeugten Gebäudeschäden sind in den Abbildungen 5.7, 5.8, 5.9 und 5.10 dargestellt. Die Ergebnisse entsprechen den Darstellungen der Test- und Evaluierungsumgebung, wobei die Möglichkeiten von Spezialeffekten wie Rauch oder Nachtsimulationen, die in X-Plane vorhanden sind, zu einer deutlichen Steigerung der Authentizität führen. Auch die Laufzeiten der Prozesskette mit X-Plane sind zu den Laufzeiten der Test- und Evaluierungsumgebung zumeist vergleichbar. Problematisch ist die Laufzeit der Auslieferung des beschädigten Gebäudes. Um die Visualisierungsmöglichkeiten von X-Plane effizient ausnutzen zu können, muss der Weg des Dateiaustausches besprochen werden. Dies stellte sich bei zunehmender Dateigröße als sogenannter *bottleneck* (Flaschenhals) der Laufzeit heraus. Da der Dateiaustausch auf dem vergleichsweise langsamen Permanent Speicher erfolgt, sind die Reaktionszeiten ab etwa 150^3 Voxeln kritisch, also länger als 5 Sekunden. Dieses Problem könnte zukünftig durch eine aufwendige Erweiterung von X-Plane umgangen werden, indem das Austauschen von Gebäudemodellen aus dem Hauptspeicher heraus zur Laufzeit der Simulation ermöglicht wird. Bei geringeren Auflösungen bleibt die Laufzeit deutlich unter den geforderten 5 Sekunden, wodurch gezeigt wird, dass der neue Ansatz zur Verwendung mit einer Simulationssoftware geeignet ist.



Abbildung 5.7: Beschädigtes Gebäude in X-Plane auf der Basis des 3D-Impactservice nach zwei Detonationen.

5.4 Bewertung des neuen Ansatzes hinsichtlich der definierten Ziele

In diesem Abschnitt werden die Möglichkeiten und Grenzen des neuen Ansatzes zur Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation hinsichtlich der definierten Ziele und Anforderungen dargestellt. Eine Übersicht der Bewertung zeigt Tabelle 5.4.

Anforderungen	3D-GIS	Realphysikalisch	3D-Videospiel	3D-Impactservice
Authentizität	-	-	+	+
Konsistenz	-	+	-	+
Reaktionszeit	-	-	+	+
Transparenz	+	+	-	+
Integrierbarkeit	-	-	-	+
Standards	+	-	-	+

Tabelle 5.4: Vergleich der Anforderungen und Ziele der bestehenden Lösungen mit dem neuen Ansatz des 3D-Impactservice (Erweiterung der Tabelle 3.2).

5.4.1 Authentizität

Bereits im Abschnitt 5.2.3 wurde gezeigt, dass der neue Ansatz die definierte Anforderung der Authentizität erfüllt. Mit der Anbindung des 3D-Impactservice an die Simulationssoftware X-Plane wurde zudem dargestellt, dass die grafische Qualität durch visuelle Effekte im Vergleich zu den Gebäudeschäden der Evaluierungsumgebung gesteigert werden kann. Die verwendete Berechnung der Schadenstransformation in Abschnitt 5.3.2 zeigt zudem, wie eine Rechenvorschrift erweitert und adaptiert werden kann. Nur durch die Definition



Abbildung 5.8: Gebäudeschaden in X-Plane auf der Basis des 3D-Impactservice. Im Hintergrund: Ein durch zwei vorherige Raketentreffer beschädigtes 3D-Gebäudemodell.

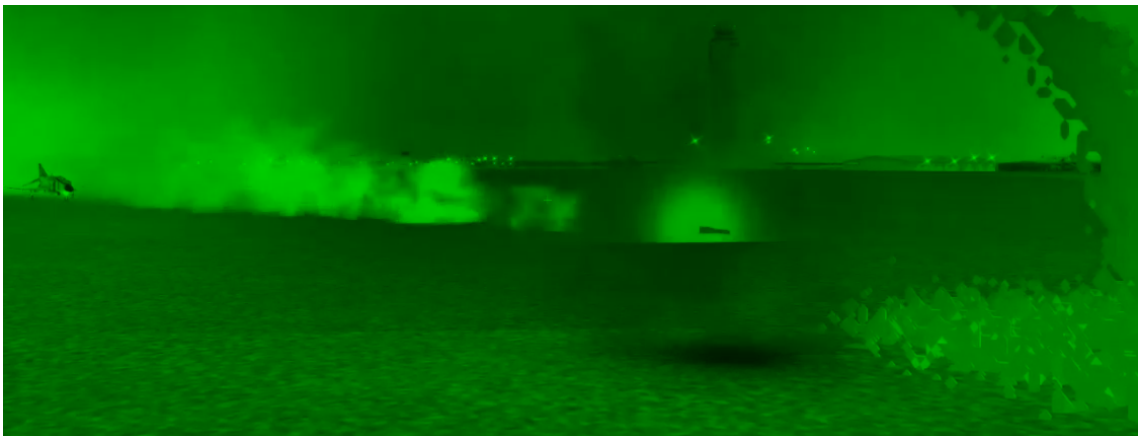


Abbildung 5.9: Gebäudeschaden in X-Plane auf der Basis des 3D-Impactservice mit speziellem grafischen Effekt (Nachtsichtmodus von X-Plane).

von drei Radien kann eine eigene Rechenvorschrift implementiert werden, die Gebäudeschäden mit der gezeigten Authentizität und visuellen Qualität erzeugt (vgl. Abbildung 5.7). Diese Erweiterbarkeit ist nicht nur wesentlich für die gezeigte Kopplung mit X-Plane, sondern ein grundsätzlicher Mehrwert des neuen Ansatzes. Der neue Ansatz bietet die Möglichkeit eigene Rechenvorschriften zur Erzeugung von Modifikationsvektorfeldern zu verwenden, die wesentlich komplexer sein können als die gezeigten Schadensstufen. Durch die Berücksichtigung der in Abschnitt 5.1.3 dargestellten Schnittstelle können diese Modifikationsvektorfelder an den 3D-Impactservice übermittelt werden.

Für die Anwendungen, die über die Tests der vorliegenden Arbeit hinaus gehen, ist es auch möglich, die unter Abschnitt 3.2.1 erwähnten *äußeren Randbedingungen* zu berücksichtigen. So könnten beispielsweise Parameter, die die aktuell simulierte Witterung betreffen, mit in die Berechnung der Schadenstransformation einfließen und die Authentizität des



Abbildung 5.10: Gebäudeschaden in X-Plane auf der Basis des 3D-Impactservice aus der Perspektive eines Flugsimulationssystems.

Gebäudeschadens weiter steigern.

Eine weitere Verbesserung der Authentizität könnte erreicht werden, indem neben der Auswertung von Geometrie und Farbe auch bildhafte Information verwendet werden. Dies wird auch im folgenden Kapitel 6.1 dargestellt.

Das eigentliche Ziel der Authentizität in der vorliegenden Arbeit, dass die Gebäudeschäden „den spezifischen Eigenschaften der Schadensursache, wie Art und Position der Waffenwirkung, soweit entsprechen, dass die Visualisierung des Gebäudeschadens für den Übenden authentisch ist“ (vgl. Abschnitt 1.2) wurde erfüllt, in dem ein Weg aufgezeigt wurde, wie die zur Laufzeit der Simulation zur Verfügung stehenden Parameter ausgewertet werden. Ferner wurde die Möglichkeit aufgezeigt und verifiziert, wie eigene Berechnungen der Schadenstransformation verwendet werden können.

5.4.2 Konsistenz

Mit dem Ziel der Konsistenz wurde gefordert, dass die „Berechnung eines Gebäudeschadens innerhalb einer verteilten Simulation eindeutig ist“ (vgl. Abschnitt 1.2). Diese Anforderung wurde erfüllt, da die Berechnung des Gebäudeschadens ausschließlich vom Modifikationsvektorfeld abhängig ist und zentral durch einen Service durchgeführt wird. Das Modifikationsvektorfeld kann innerhalb einer verteilten Simulation eindeutig definiert werden und die Berechnung des Gebäudeschadens wird zentral durch den 3D-Impactservice ausgeführt.

Darüber hinaus ging mit dem Ziel der Konsistenz die Anforderung einher, dass unterschiedliche Repräsentationsformen (beispielsweise unterschiedliche LoD eines Gebäudes oder unterschiedliche Repräsentationen eines Gebäudes aus verschiedenen Simulationssystemen) konsistent verändert werden. Diese Anforderung wurde erfüllt, indem beispielsweise in Abbildung 5.4 gezeigt wird, wie ein Modifikationsvektorfeld, bei Variation der Auflösung von 3D-Gebäudemodell und Modifikationsvektorfeld, auf ein unbeschädigtes Gebäude ange-

wendet werden kann.

5.4.3 Transparenz

In Abschnitt 5.1.3 wurde gezeigt, wie unter der Verwendung von Geodatenstandards eine transparente Schnittstelle zur Erzeugung von Gebäudeschäden definiert wird. Durch diese Schnittstelle können Modifikationsvektorfelder zur Erzeugung von Gebäudeschäden definiert werden, was der geforderten Funktionalität einer API gleich kommt.

Darüber hinaus wurden mit der vorliegenden Arbeit alle Prozessschritte des neuen Ansatzes dokumentiert und beschrieben, wodurch proprietäre Methoden vermieden und der Anforderung der Transparenz zusätzlich nachgekommen wird.

5.4.4 Reaktionszeit

In den Abschnitten 5.2.3 und 5.3.3 wurde ausführlich erläutert, dass der neue Ansatz die geforderte maximale Reaktionszeit von 5 Sekunden nicht überschreitet. Die erzielten Reaktionszeiten sind stets in Relation zur verwendeten Hardware zu bewerten. Da die Tests auf einer ca. vier Jahre alten Hardware durchgeführt wurden, sind mit einem entsprechend aktuelleren Systemen kürzere Reaktionszeiten zu erwarten.

5.4.5 Integrierbarkeit

Das der 3D-Impactservice „als Modul in unterschiedliche verteilte Simulationen mit variierenden Simulationssystemen“ (vgl. Abschnitt 1.2) eingesetzt werden kann, zeigt bereits seine Implementierung als Service konform zum MSaaS Konzept. Der 3D-Impactservice ist keine Erweiterungskomponente dedizierter Sicht- oder Simulationssysteme, sondern eine eigenständige Software. Der 3D-Impactservice kann flexibel in beliebige Joint-Exercises eingebunden werden.

5.4.6 Bestehende Standards

Das Ziel der Berücksichtigung bestehender Standards wird sowohl bezüglich der 3D-Gebäudemodelle, als auch bezüglich der verteilten Simulation erfüllt. Im Abschnitt 5.1.1 wird gezeigt, dass der verwendete Ansatz der Voxelisierung so gewählt wird, dass die Berücksichtigung aller aufgezeigten Normen und Standards der 3D-Gebäudemodelle (SEDRIS, CityGML und CDB mit OpenFlight vgl. Abschnitt 2.6) möglich ist. Der 3D-Impactservice ist für die gängigen Normen und Standards der 3D-Gebäudemodelle in der virtuellen Simulationen verwendbar.

Die Normen und Standards der verteilten Simulation werden berücksichtigt, indem der 3D-Impactservice sich an den beiden gängigen Standards der verteilten Simulation (HLA und DIS) orientiert. Durch die Verwendung der *Munition Detonation Interaction* des RPR-FOM als Vorlage der codierten Nachricht zwischen X-Plane und 3D-Impactservice wird

gezeigt, dass mit dem 3D-Impactservice und, unter der Verwendung etablierter Standards, die Erzeugung von Gebäudeschaden möglich ist.

Zusätzlich zu den Normen und Standards wurden etablierte Techniken wie das MSaaS Konzept im 3D-Impactservice berücksichtigt.

Kapitel 6

Schlussbemerkungen

In diesem finalen Kapitel werden in einem Ausblick die Themenfelder, Aspekte und Perspektiven aufgezeigt, die Inhalt zukünftiger Forschungsaktivitäten im Themenbereich der Gebäudeschäden sein könnten. Auch die Grenzen des neuen Ansatzes werden dargestellt.

6.1 Grenzen des Ansatzes und Ausblicke

6.1.1 Authentizität

In den voran gegangenen Kapiteln wurde gezeigt, dass mit dem neuen Ansatz eine große Bandbreite von Gebäudeschäden erzeugt werden kann. Mit dieser großen Bandbreite gehen allerdings auch Möglichkeiten einher, dass Gebäudeschäden entstehen, die nicht authentisch sind. Zwar ist es mittels der API möglich authentische Gebäudeschäden zu erzeugen, allerdings kann der 3D-Impactservice keine Authentizität garantieren. Die API erlaubt auch die Erzeugung von offensichtlich nicht authentischen Gebäudeschäden. Beispielsweise könnte ein Modifikationsvektorfeld alle Wände eines Gebäudes entfernen, das Dach des Gebäudes jedoch unberührt lassen. In der Folge würde ein nicht authentischer Gebäudeschaden entstehen, der ein schwebendes Dach darstellt.

Dieses Phänomen stellt eine Grenze des neuen Ansatzes dar, da mit dieser Arbeit nur die geometrischen Modifikation berücksichtigt werden. Die Berücksichtigung von logischen oder statischen Abhängigkeiten, wie einem schwebenden Dach, waren kein Bestandteil dieser Arbeit und sind auch kein Bestandteil des 3D-Impactservice.

Allerdings wäre die Erweiterung des 3D-Impactservice für logische oder statische Abhängigkeiten durchaus möglich. Auch wenn die Abhängigkeiten nicht per se in den Gebäudeschaden mit einfließen, sind Möglichkeiten zur Berücksichtigung der Statik denkbar. So wäre beispielsweise ein dem 3D-Impactservice nachgelagerter Prozess zur statischen Verifikation des Gebäudeschadens im Anschluss an die Retriangulierung realisierbar. Innerhalb des Prozesses müsste jedes Dreieck des Gebäudeschadens auf physikalische oder statische Authentizität geprüft und gegebenenfalls korrigiert werden.

Ob ein derartiger Prozess, der das einzelne Gebäude betrachtet, nicht der Anforderung der Konsistenz entgegen steht, wird Bestandteil zukünftiger Untersuchungen sein.

6.1.2 Reduktion der Anzahl der Dreiecke des Ausgangszustandes

Ein mögliches Problem der bisherigen Implementierungen des 3D-Impactservice ist die hohe Anzahl der Dreiecke des beschädigten Gebäudes. Durch die Verwendung des Marching Cube Algorithmus ist die Entstehung diverser koplanarer Dreiecke im Ausgangszustand unvermeidbar. Zwar führte die Anzahl der Dreiecke weder in der Evaluierungsumgebung noch in der Anbindung an X-Plane zu kritischen Laufzeiten, dennoch sollte für einen Produktivbetrieb des 3D-Impactservice eine Strategie zur Reduktion der Dreiecke entwickelt werden.

Eine mögliche Strategie wäre die Verwendung eines nachgelagerten Dreiecks-Reduktions-Prozesses. Das beschädigte Gebäude könnte in einem Postprocessing die Dreiecke auf Koplanarität prüfen und entsprechend optimieren. Zu diesem Vorgang der sogenannten *Mesh-Optimization* existieren eine Vielzahl an Lösungen sowie kommerziellen Softwareprodukten. Eine Evaluierung, welcher Optimierungsalgorithmus von welcher Software unter den Anforderungen der Handhabung der Gebäudeschäden einsetzbar ist, wird Bestandteil von Weiterentwicklungen für einen Produktivbetrieb des 3D-Impactservice sein.

6.1.3 3D-Impactservice auf Basis von Geodatenstandards

Im Abschnitt 2.7 wurden Lösungen erwähnt, wie Normen und Standards der Geoinformation in der verteilten, virtuellen Simulation integriert werden können. Einige Vorteile dieser Geodatenstandards wurde in dieser Arbeit mit der Formalisierung einer Schnittstelle für Modifikationsvektorfelder genutzt. Diese Verwendung stellt jedoch nur einen geringen Teil der Möglichkeiten von Geodatenstandards dar. In [6, 44, 45, 82] werden komplexe Datenhaltungskonzepte und serviceorientierte Schnittstellen vorgestellt sowie deren Anwendbarkeit zur Datenversorgung von Simulationssystemen vor der Simulation verifiziert. Dies führt auf die Frage, ob nicht auch der 3D-Impactservice konform zu etwaigen Geodatenstandards implementiert werden könnte.

Ein möglicher OGC basierter Standard wäre der Web Processing Service. „Der Web Processing Service (WPS) ist ein Mechanismus, um über das Internet eine räumliche Analyse von Geodaten durchzuführen“ [95]. Allerdings ist, nach Michael und Ames [55], XML für jede Arte der Kommunikation mit dem WPS zu verwenden (original: „The specification indicates that extensible markup language (XML) should be used for all communication.“ [55]). Ob unter dieser XML-Pflicht und im Hinblick auf die geforderte Reaktionszeit ein OGC konformer WPS in der verteilten, virtuellen Simulationen verwendet werden kann, wird untersucht werden müssen.

6.1.4 Erweiterte Anwendbarkeit auf Basis der VIntEL-Architektur

Der bisherige Stand der Technik wurde im Abschnitt 2.7 auf der Basis des Forschungsprojekts VIntEL aufgezeigt. Die Handhabung von Gebäudeschäden im Projekt VIntEL ist in Abbildung 2.15 dargestellt. Basierend auf diesem Stand der Technik wird in Abbildung 6.1 demonstriert, dass der 3D-Impactservice in diese Architektur integriert werden könnte.

Anders als der Austausch von prozentualen Schadensklassifikationen, ermöglicht der 3D-Impactservice die Berechnung von Gebäudeschäden zur Laufzeit der Simulation unter Berücksichtigung des Schadensereignisses. Außerdem wird die Auslieferung der beschädigten 3D-Gebäudemodelle an die Simulationssysteme skizziert.

In Abbildung 6.1 ist auch die Anbindung der Effektservices an den 3D-Impactservice dargestellt. Wie die zumeist kommerziellen Effektservices (vgl. Abschnitt 2.7) erweitert werden könnten, um an Stelle der prozentualen Schadensklassifikation ein Modifikationsvektorfeld zu liefern, wird in der vorliegenden Arbeit nicht thematisiert. Mit dem 3D-Impactservice wird ein Weg aufgezeigt, wie eine Erzeugung von Gebäudeschäden möglich ist. Die Umsetzung der Fachspezifika von Waffenwirkungen und Waffensystemen muss durch die Hersteller bisheriger Effektservices erfolgen und ist Bestandteil zukünftiger Umsetzungs- und Forschungsaktivitäten.

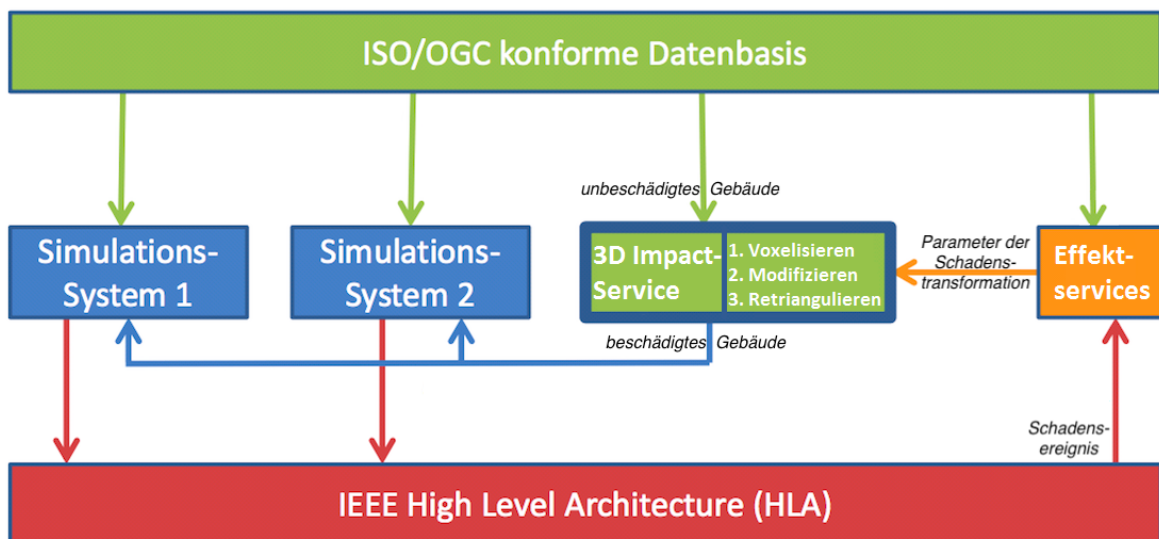


Abbildung 6.1: Integration des 3D-Impactservice in die Architektur einer verteilten, virtuellen Simulation zur Handhabung von dynamischen Gebäudeschäden.

Mit Abbildung 6.1 wird darüber hinaus verdeutlicht, dass die bisher als Stand der Technik geltende *Level 3* Interoperabilität (vgl. LCIM in Abschnitt 2.6) durch die Verwendung des 3D-Impactservice gesteigert werden kann. Im bisherigen Stand der Technik ermöglicht der Umgang mit Gebäudeschäden in der verteilten Simulation nur ein gleiches Verständnis von Information [52], was hinreichend für *Level 3* ist. „Welche Konsequenz sich beim empfangenden Föderaten aus einer berechneten Trefferwirkung ergibt, ist jedoch Frage des pragmatischen Levels“ [52].

Da der 3D-Impactservice exakt diese Trefferwirkung, nämlich einen konkretes beschädigtes

Gebäude, berechnet und zur Visualisierung bereitstellt, ist der hier vorgestellte Ansatz zur Handhabung von Gebäudeschäden ein Beispiel zur Realisierung von *Level 4* (vgl. 2.6 pragmatische Interoperabilität) nach der LCIM Klassifikation für Interoperabilität.

6.1.5 Berücksichtigung von Texturen

In den bereits mehrfach erwähnten zukünftigen Umsetzungs- und Forschungsaktivitäten können auch technische Details einfließen, die in der vorliegenden Arbeit zur Reduktion des Problems nicht thematisiert wurden. Ein Beispiel für ein technisches Detail ist die Verwendung von Texturen im unbeschädigten 3D-Gebäudemodell. Bisher ist der 3D-Impactservice auf die Verwendung von farblichen Dreiecken beschränkt. Bildhafte Informationen der 3D-Gebäudemodelle werden nicht berücksichtigt. Es ist zu untersuchen, wie diese Texturen mittels Voxelisierung erhalten bleiben und wie die Texturen durch etwaige Modifikationen zur Anzeige gebracht werden könnten. Auch die Möglichkeit der farblichen Modifikation durch einen Gebäudeschaden, beispielsweise die Rußbildung auf dem Mauerwerk, ist ein technisches Detail zukünftiger Untersuchungen.

6.2 Fazit

Es ist offensichtlich, dass die gezeigten Gebäudeschäden einem zur Laufzeit der virtuellen verteilten Simulation auftretenden Schadensereignis entsprechen können. Dies stellt den Mehrwert des Ansatzes gegenüber dem bisherigen Austausch von prozentualen Schadensklassifikationen per LoS-Konzept dar. Da mittels 3D-Impactservice ein beschädigtes Gebäude dem Schadensereignis der verteilten Simulation entsprechen kann, ist die Erzeugung eines authentischen Feedbacks des Waffeneinsatzes für den Übenden möglich. Damit wird „die Handhabung von Gebäudeschäden in der verteilten, virtuellen Simulation verbessert“ (vgl. Ziel dieser Arbeit in Abschnitt 1.1).

Literaturverzeichnis

- [1] 3D-STADTMODELLE: *Informatiosbroschüre zur Nutzung von virtuellen 3D-Stadtmodellen der gemeinsamen Kommission 3D-Stadtmodelle der DGPF und DGfK*. 2013. – URL <http://www.ingeoforum.de/3d-stadtmodelle.pdf>. – letzter Zugriff: 07.05.2018
- [2] ADV-CITYGML-PROFILE: *AdV-CityGML-Profil für 3D-Gebäudemodelle Ergebnisse der PG „3D-Gebäudemodelle“ der Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (Adv)*. 2013. – URL <http://www.adv-online.de>. – letzter Zugriff: 07.05.2018
- [3] ALI, Samia A. ; HUSSAIN, Khaled F. ; SAYED, Ahmed M.: *Fast Generation of Demolition Special Effects on 3D Buildings*. In: *Journal of Engineering Sciences, Assiut University*. Vol. 39, No 1. 2011, S. 49-71
- [4] AUTOBILD: *Internetpräsenz der Zeitschrift Autobild..* – URL <http://www.autobild.de>. – letzter Zugriff 07.05.2018
- [5] AVAL ; HARTMANN, Mats (Hrsg.) ; ALVÍ, Peter (Hrsg.) ; PELO, Johan (Hrsg.) ; WALLSTRÖM, Dag (Hrsg.): *Assessment of Vulnerability And Lethality (AVAL) User's manual*. 2012. – URL <https://www.tno.nl/media/1707/tno-vulnerability-and-lethality2.pdf>. – letzter Zugriff: 07.05.2018
- [6] AVERDUNG, Christoph: *Direkte Verwendung von ISO/OGC-konformen 3D-Geodaten (CityGML) in der Simulation*. In: *Proceedings of Geoforum MV 2010*. Rostock-Warnemünde, 2010, S. 27–30
- [7] BILDSTEIN, Frank: *Nutzung von Geodaten zur Visualisierung geospezifischer Übungsszenarien in Trainingssimulatoren. Fachvortrag auf dem Workshop 3D-Stadtmodelle der DGfK und der DGPF von F. Bildstein, Rheinmetall Defense*. 2014. – URL http://www.3d-stadtmodelle.org/3d-stadtmodelle_2014/vortraege/04_Bildstein_Geodatennutzung_Trainingssimulatoren.pdf. – letzter Zugriff: 07.05.2018

- [8] BUNDESHEER: *Internetpräsenz des Deutschen Heeres*. – URL <http://www.deutschesheer.de>. – letzter Zugriff: 07.05.2018
- [9] BUNDESREGIERUNG: *Bedeutung von Videospiele für Öffentlichkeitsarbeit und Personalwerbung der Bundeswehr. Antwort der Bundesregierung auf die Kleine Anfrage der Abgeordneten Ulla Jelpke, Harald Koch, Petra Pau, weiterer Abgeordneter und der Fraktion DIE LINKE*. 2011. – URL <http://dipbt.bundestag.de/dip21/btd/17/075/1707599.pdf>. – letzter Zugriff 07.05.2018
- [10] BUNDESREGIERUNG: *Vorstellung des Zentrums für Geoinformationswesen der Bundeswehr (ZGeoBw) auf der Webpräsenz der Bundesregierung*. 2015. – URL https://www.bundesregierung.de/Webs/Breg/DE/Themen/Forschung/ressort/ZGeoBw/_node.html. – letzter Zugriff: 07.05.2018
- [11] CAE: *Internetpräsenz der Canadian Aviation Electronics Inc. (CAE)*. – URL <http://www.cae.com>. – letzter Zugriff 07.05.2018
- [12] CASSIDIAN: *Internetpräsenz der European Aeronautic Defence and Space Company (EADS) Deutschland Division Cassidian GmbH. Seit 2014 Airbus Defence and Space Division der Airbus Group AG*. – URL <https://airbusdefenceandspace.com>. – letzter Zugriff 18.08.2016
- [13] CDB: *Open Geospatial Consortium, Common Database Specification (CDB)*. Onlinedokument. 2016. – URL <http://www.opengeospatial.org/standards/cdb>. – letzter Zugriff: 07.05.2018
- [14] CITYGML: *Internetpräsenz der City Geography Markup Language (CityGML)*. – URL <http://www.citygml.org/>. – letzter Zugriff 07.05.2018
- [15] CPA: *Internetpräsenz der CPA ReDev GmbH*. – URL <http://www.cpa-redev.de>. – letzter Zugriff 07.05.2018
- [16] DESEEP: *IEEE 1730-2010 - Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)*. 2011
- [17] DIEHL: *Internetpräsenz der Diehl Defence GmbH*. – URL <http://www.diehl.com>. – letzter Zugriff: 07.05.2018
- [18] DIS: *IEEE Std 1278.1-1995: Standard for Distributed Interactive Simulation (DIS) Application Protocols*. 2012. – URL <http://ieeexplore.ieee.org/document/6387564/>. – letzter Zugriff: 07.05.2018
- [19] EGENHOFER, Max J.: A model for detailed binary topological relationships. In: *Geomatica* 34 (1993), S. 261–273

- [20] FE: *Open Geospatial Consortium, Filter Encoding (FE)*. Onlinedokument. 2014. – URL <http://www.opengeospatial.org/standards/filter>. – letzter Zugriff: 07.05.2018
- [21] FLUGSIMULATOREN: *Privates Internetportal mit Vergleich und Bewertung ziviler Flugsimulatoren im deutschsprachigen Raum*. 2017. – URL <http://www.flugsimulator-vergleich.de>. – letzter Zugriff: 07.05.2018
- [22] FRIEDRICHS, Hauke ; FUCHS, Christian: *Die simulierte Armee*. In: *Die Zeit-Online*. 2015. – URL <http://www.zeit.de/politik/deutschland/2015-08/bundeswehr-ausbildung-simulator-verteidigungsministerium/komplettansicht>. – letzter Zugriff: 07.05.2018
- [23] GABLER: *Gabler Wirtschaftslexikon. Stichwort: Simulation*. Gabler Verlag. – URL <http://wirtschaftslexikon.gabler.de>. – letzter Zugriff: 07.05.2018
- [24] GEOINFODOK: *Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok)*. Onlinedokument. 2008. – URL <http://www.adv-online.de>. – letzter Zugriff: 07.05.2018
- [25] GEOINFORMATIKLEXIKON ; BILL, Ralf (Hrsg.): *Online Lexikon der Professur für Geodäsie und Geoinformatik der Universität Rostock*. 2016. – URL <http://www.geoinformatik.uni-rostock.de/lexikon.asp>. – letzter Zugriff: 07.05.2018
- [26] GLOSE, Carsten: *VIntEL- Eine verteilte Simulationslandschaft, CPM-Forum "Joint Operations"05.2015*. – URL <http://cpm-st-augustin.de/magazine/2015/6-2015/#/44/>. – letzter Zugriff: 07.05.2018
- [27] GLOSSARY: *United States Department of Defense Modeling and Simulation Glossary (Stichwortverzeichnis für Modellierung und Simulation des Verteidigungsministeriums der USA)*. – URL <http://www.acqnotes.com/Attachments/DoD%20M&S%20Glossary%201%20Oct%2011.pdf>. – letzter Zugriff 07.05.2018
- [28] GML: *Open Geospatial Consortium, Geography Markup Language (GML)*. Onlinedokument. 2012. – URL <http://www.opengeospatial.org/standards/gml>. – letzter Zugriff: 07.05.2018
- [29] GOOGLE EARTH: *Software zur Darstellung eines virtuellen Globus mit Luftbildern, Satellitenaufnahmen und 3D-Objektmodellen der Firma Google*. 2004. – URL <http://earth.google.de>. – letzter Zugriff: 07.05.2018

- [30] GRIFFIN, Joshua W.: *Experimental and Analytical Investigation of Progressive Collapse through Demolition Scenarios and Computer Modeling*, Diplomarbeit, 2008
- [31] GUSTAVSON, Paul ; CHASE, Tram ; ROOT, Larry ; CROSSON, Karl: Moving towards a Service-Oriented Architecture (SOA) for Distributed Component Simulation Enviroments. In: *Proceedings of the 2005 Spring Simulation Interoperability Workshop.*, 2005
- [32] HLA: *IEEE 1516.3-2003 - Recommended Practice for High Level Architecture Federation Development and Execution Process (FEDEP)*. 2003
- [33] HLA: *IEEE 1516.4-2007 - Recommended Practice for Verification, Validation, and Accreditation of a Federation an Overlay to the High Level Architecture Federation Development and Execution Process*. 2007
- [34] HLA: *IEEE 1516-2010: Standard for Modeling and Simulation High Level Architecture - Framework and Rules*. 2010
- [35] HLA: *IEEE 1516.1-2010 - Standard for Modeling and Simulation High Level Architecture - Federate Interface Specification*. 2010
- [36] HLA: *IEEE 1516.2-2010 - Standard for Modeling and Simulation High Level Architecture - Object Model Template (OMT) Specification*. 2010
- [37] HOFMANN, Marko A.: Essential preconditions for coupling model-based information systems. In: *Proceedings of NATO M&S Group (NMSG) Conference on C3I and M&S Interoperability*. Antalya, Turkey, October 2003
- [38] IABG: *Internetpräsenz der Industrieanlagen-Betriebsgesellschaft mbH (IABG)*. – URL <http://www.iabg.de>. – letzter Zugriff 07.05.2018
- [39] KDB: *Konzeption der Bundeswehr, Fü S VI 2 - Az 09-02-04/V5-NfD*. 2004
- [40] KHAYARI, Rachid El A.: *Simulations- und Testumgebung der Bundeswehr (SuTBw), Institut für Technik Intelligenter Systeme (ITIS) der Universität der Bundeswehr München*. 2007. – URL <https://www.unibw.de/itis/sut>. – letzter Zugriff: 19.08.2016
- [41] KMW: *Internetpräsenz der Krauss-Maffei Wegmann GmbH & Co. KG (KMW)*. – URL <http://www.kmweg.de>. – letzter Zugriff 07.05.2018
- [42] KOLBE, Thomas H. ; GRÖGER, Gerhard ; PLÜMER, Lutz: CityGML - Interoperable Access to 3D City Models. In: *Proceedings of the 1st International Symposium on Geoinformation for Disaster Management*. Delft, The Netherlands, März 2005

- [43] KRÜCKHANS, Martin: *Ein Detektor für Ornamente auf Gebäudefassaden auf Basis des histogram-of-oriented-gradients- Operators*. 2010. – URL http://www.ikg.uni-bonn.de/uploads/tx_ikgpublication/krueckhans2010detektor.pdf. – Masterarbeit, letzter Zugriff: 18.08.2016
- [44] KRÜCKHANS, Martin: *SEDRIS als Datenmodell für eine synthetische 3D-Umweltdatenbasis. Fachvortrag auf dem Workshop 3D-Stadtmodelle der DGfK und der DGPF von Martin Krückhans, CPA Systems*. 2011. – URL http://www.3d-stadtmodelle.org/3d-stadtmodelle_2011/vortraege/09_Krueckhans_SEDRIS_Umweltdatenbasis.pdf?PHPSESSID=c5bb87ab6350ae4f37da76685cc83c91. – letzter Zugriff: 07.05.2018
- [45] KRÜCKHANS, Martin: ISO ans OGC compliant database technology for the development of simulation object databases. In: *Proceedings of 2012 Winter Simulation Conference, 2012*
- [46] KRÜCKHANS, Martin: CityGML in Simulationsumgebungen. In: *Kartographische Nachrichten 2/3-13 (2013), April, S. 92–94*
- [47] KRÜCKHANS, Martin: *ISO/OGC-konforme Datenbanktechnologie für den Synthetic Environment Service (SES) in der verteilten Simulation (CPA Systems GmbH). Fachvortrag auf dem Forum Modellbildung und Simulation der deutschen Gesellschaft für Wehrtechnik (DWT) in Bonn Bad Godesberg*. 2013. – URL https://www.unibw.de/itis/archiv/website_2014/aktuelles/veranstaltungen/6.%20Workshop/at_download/down1. – letzter Zugriff: 19.08.2016
- [48] LEE, Eun-Jin ; EL-TAWIL, Sherif: FEMvrml: An interactive virtual environment for visualization of finite element simulations results. (2008)
- [49] LORENSEN, Willam E. ; CLINE, Harvey E.: *Marching Cubes: A high resolution 3D surface construction algorithm*. In: *Computer Graphics, Vol. 21, Nr 4*. 1987, S. 163-169
- [50] LUFTWAFFE: *Internetpräsenz der Deutschen Luftwaffe*. – URL <http://www.luftwaffe.de>. – letzter Zugriff: 07.05.2018
- [51] LÜBBERS, Hubertus: *VIntEL, Quo vadis? (BAAIN Bw P2.3) Vortrag auf dem ITIS Workshop "Komponenten und Virtualisierung in der Modellbildung und Simulation", Universität der Bundeswehr München*. 2013. – URL https://www.unibw.de/itis/archiv/website_2014/aktuelles/veranstaltungen/6.%20Workshop/at_download/down1. – letzter Zugriff: 19.08.2016

- [52] LÜTHI, Johannes: *Fair Fight in verteilter Simulation - Dienste als Allheilmittel? Vortrag auf dem ITIS-Workshop "Qualitätssteigerung durch Standardisierung?!", Universität der Bundeswehr München*. 2011. – URL https://www.unibw.de/itis/archiv/website_2014/aktuelles/veranstaltungen/archiv/4workshop-perspektiven-der-modellbildung-und-simulationen-17.-18.-januar-2011/at_download/down1. – letzter Zugriff: 18.08.2016
- [53] MASTERPLAN: *Modelling and Simulation Masterplan, Department of Defence, DoD 5000.59-P*. 1995
- [54] MATTERN, Steffen ; BLANKENHORN, Gunther ; SCHWEIZERHOF, K: Numerical simulation of controlled building collapse with finite elements and rigid bodies. Case studies and validation. In: *ECCOMAS Thematic Conference in Structural Dynamics and Earthquake Engineering, Kreta, Griechenland, 2007*
- [55] MICHAEL, Christopher ; AMES, Daniel P.: Evaluation of the OGC Web Processing Service for Use in a Client-Side GIS. (2007)
- [56] MONDESIRE, S. C. ; MAXWELL, D. B. ; SZIELINSKI, S. ; MARTIN, G. A.: *Physics Engine Benchmarking in Three-Dimensional Virtual World Simulation. MOSDSIM World*. 2016. – URL http://www.modsimworld.org/papers/2016/Physics_Engine_Benchmarking_in_Three-Dimensional_Virtual_World_Simulation.pdf. – letzter Zugriff: 07.05.2018
- [57] MONTANI, C. ; SCATENI, R. ; SCOPIGNO, R.: *Discretized Marching Cubes*. In: *Visualization '94 Proceedings, IEEE Computer Society Press*. 1994, S. 281-287
- [58] MÄNTYLÄ, Martti: *An Introduction to Solid Modeling.Principles of Cuomputer Science. Computer Science Press, Maryland, USA*. 1988
- [59] NEUGEBAUER, Eckehard ; NITSCH, Daniel ; HENNE, Oliver: Architecture for a Distributed Integrated Test Bed. In: *NATO RTO Modelling and Simulation Group Symposium (MSG-069), Brüssel, Belgien, 2009*
- [60] NOKIAMAPS: *Software zur Darstellung eines virtuellen Globus mit Luftbildern, Satellitenaufnahmen und 3D-Objektmodellen der Firma Nokia*. 2011. – URL <http://here.com>. – letzter Zugriff: 07.05.2018
- [61] OPENFLIGHT: *Presagis OpenFlight Scene Specification for Version 16.4*. Onlinedokument. 2008. – URL https://www.presagis.com/workspace/uploads/files/openflight14-2_15-0.pdf. – letzter Zugriff: 07.05.2018

- [62] PAWLASZCZYK, Dirk: *Skalierbare agentenbasierte Simulation*, Universität Ilmenau, Dissertation, 2009
- [63] PROTZMANN, Michael: *Einsatz und Nutzen von Fachdiensten in verteilter Simulation (IABG mbH), Vortrag auf dem ITIS Workshop "Komponenten und Virtualisierung in der Modellbildung und Simulation"*, Universität der Bundeswehr München. 2013. – URL https://www.unibw.de/itis/archiv/website_2014/aktuelles/veranstaltungen/6.%20Workshop/at_download/down1. – letzter Zugriff: 19.08.2016
- [64] RDE: *Internetpräsenz der Rheinmetall Defence Electronics GmbH (RDE)*. – URL <http://www.rheinmetall-defence.com>. – letzter Zugriff: 07.05.2018
- [65] ROSSMANN, Jürgen ; SCHLUSE, Michael ; WASPE, Rals ; HOPPEN, Martin: Real Time Capable Data Management Architecture for Database-Driven 3D Simulation Systems. In: *Proceedings of the 22nd international conference on Database and expert systems applications (DEXA 2011)* Bd. 1. Toulouse, Frankreich, 2011, S. 262–269
- [66] RPRFOM: *SISO-STD-001.1-1999: Real-time Platform Reference Federation Object Model (RPR FOM 1.0)*. 1999
- [67] SCHMITTWILKEN, Jörg: *Attributierte Grammatiken zur Rekonstruktion und Interpretation von Fassaden*, Universität Bonn, Dissertation, 2012
- [68] SCHWARZ, Michael ; SEIDEL, Hans-Peter: *Fast parallel surface and solid voxelization on GPUs*. In: *ACM Transactions on Graphics (TOG)*. 2010. – URL <http://research.michael-schwarz.com/publ/files/vox-siga10.pdf>. – letzter Zugriff: 07.05.2018
- [69] SEDRIS: *Synthetic Environment Data Representation and Interchange Specification*. Onlinedokument. 2012. – URL <http://standards.sedris.org>. – letzter Zugriff: 07.05.2018
- [70] SIEGFRIED, Robert ; BERTSCHIK, Michael ; HAHN, Matthias ; HERRMANN, Günter ; LÜTHI, Johannes ; ROTHER, Martin: *Effective and Efficient Training Capabilities through Next Generation Distributed Environment*. In: *Proceedings of NATO Modelling and Simulation Group (NMSG) Multi-Workshop, Sydney, Australia*. 2013
- [71] SIEGFRIED, Robert ; LAUX, Alexander ; HERRMANN, Günter ; LÜTHI, Johannes: *VEVA - A Procedure Model for Distributed Simulation*

- Experiments. In: *Proceedings of NATO RTO Modeling and Simulation Group Symposium (NMSG-076)*. Utrecht, Niederlande, 2010
- [72] SIEGFRIED, Robert ; LÜTHI, Johannes ; HERRMANN, Günter ; HAHN, Matthias: How to ensure Fair Fight in LVC Simulation: Architectural and Procedural Approaches. In: *Proceedings of NATO RTO Modeling and Simulation Group Symposium (MSG-087)*. Bern, Schweiz, Oktober 2011
- [73] SIG3D: *Internetpräsenz der Special Interest Group 3D (SIG3D)*. – URL <http://www.sig3d.org>. – letzter Zugriff 07.05.2018
- [74] SIKIWAT, Tanongsak ; BREIDT, Michael ; HARTMANN, Dietrich: Collapse Simulations of Large Complex Structures Due to Controlled Explosives. In: *Proceedings 7th EUROMECH Solid Mechanics Conference*, 2009
- [75] SIMUTECH: *Internetpräsenz der Gesellschaft für Fahrsimulation mbH*. – URL <http://www.simutech.de>. – letzter Zugriff 07.05.2018
- [76] SISO: *Internetpräsenz der Simulation Interoperability Standards Organization (SISO)*. – URL <http://www.sisostds.org/>. – letzter Zugriff 07.05.2018
- [77] SISO: *SISO-REF-010-2017: Reference for Enumerations for Simulation Interoperability*. 2017. – URL https://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=46171. – letzter Zugriff: 07.05.2018
- [78] SPATIALSCHEMA: *ISO 10109-2007: Geographic Information - Spatial Schema*. 2007
- [79] STROBEL, Stefan ; REINHARDT, Wolfgang: Die Bedeutung von Profilen für den Datenaustausch mittels Geography Markup Language. In: *Mitteilungen des Bundesamtes für Kartographie und Geodäsie* Bd. 39, 2007
- [80] STÜBER, Ralf: *Generalisierung von Gebäudemodellen unter Wahrung der visuellen Richtigkeit*, Universität Bonn, Dissertation, 2004
- [81] STÜBER, Ralf: *Der Synthetic Environment Service (SES) in der verteilten Simulation (CPA ReDev)*. Fachvortrag auf dem 9. ITIS Workshop "Perspektiven der Modellbildung und Simulation", Universität der Bundeswehr München. 2016. – URL https://www.unibw.de/itis/aktivitaeten/veranstaltungen/perspektiven/copy_of_ws_ms_2015/at_download/down3. – letzter Zugriff: 02.09.2016
- [82] STÜBER, Ralf ; KRÜCKHANS, Martin: Increased sustainability of Simulation Object Databases using international norms and standards. In: *Proceedings*

- of Symposium on Transforming Defence through Modelling and Simulation Opportunities and Challenges*. Schweden, 2012
- [83] SUKUMAR, N ; BOLANDER, J. E.: *Voronoi-based interpolants for fracture Modelling*. Tesselations in the Sciences, 2009
- [84] TENA: *Test and Training Enabling Architecture (TENA)*. 2012. – URL <https://www.tena-sda.org>. – letzter Zugriff: 07.05.2018
- [85] THALES: *Deutsche Internetpräsenz der Thales Group*. – URL <http://www.thalesgroup.com/germany>. – letzter Zugriff: 07.05.2018
- [86] TKM&S: *Teilkonzeption Modellbildung und Simulation in der Bundeswehr (TK M&S Bw), Generalinspekteur der Bundeswehr, IT5 - Az 09-02-05/VS-NfD*. 2006. – VS-NfD
- [87] TOLK, Anderas ; DIALLO, Saikou Y. ; KING, Robert D. ; TURNITSA, Charles D. ; PADILLA, Jose J.: *Conceptual modeling for composition of model-based complex systems*. CRC Press, 2010
- [88] TOLK, Andreas: *Engineering Principles of Combat Modeling and Distributed Simulation*. John Wiley & Sons, 2012
- [89] TOLK, Andreas ; MUGUIRA, James A.: The Levels of Conceptual Interoperability Model. In: *Proceedings of Fall Simulation Interoperability Workshop*. Orlando, Florida, 2003
- [90] TURNITSA, Charles: Extending the levels of conceptual Interoperability model. In: *Proceedings IEEE summer computer simulation conference, IEEE CS Press*, 2005
- [91] UFER, Hartmut ; DERLICH, Helmut ; NEUGEBAUER, Eckehard ; KAPPEN, Klaus: Abschlussbericht zur Studie: Konzept Systemdemonstrator Verteilte Integrierte Erprobungslandschaft (Konzept SD VIntEL) / IABG mbH, Deutsche Bundeswehr. 2009 (E/UR2E/8A107/7F193). – Forschungsbericht
- [92] VBS: *Interpräsenz der Simulationsoftware Virtual Battlespace (VBS) der Firma Bohemia Interactive Simulations*. – URL <https://bisimulations.com/virtual-battlespace-3>. – letzter Zugriff: 07.05.2018
- [93] VTMÄK: *Internetpräsenz der Softwarefirma VT MÄK (u.a. Hersteller einer kommerziellen HLA RTI Implementierung)*. – URL <https://www.mak.com>. – letzter Zugriff 07.05.2018
- [94] WFS: *Open Geospatial Consortium, Web Feature Service 2.0 Interface Standard (WFS)*. Onlinedokument. 2014. – URL

<http://www.opengeospatial.org/standards/wfs>. – letzter Zugriff: 07.05.2018

- [95] WIKIPEDIA: *Online Enzyklopädie Wikipedia..* – URL <http://www.wikipedia.de>. – Hinweis: Die Artikel der Wikipedia-Enzyklopädie werden von den Usern selbst erstellt, korrigiert und aktualisiert. Eine qualifizierte wissenschaftliche Rezension oder Überprüfung von Artikeln erfolgt jedoch nicht. Durch die stetige Kontrolle der Artikel durch die User, kann jedoch eine gewisse Allgemeingültigkeit im Sinne einer weit verbreiteten Meinung angenommen werden. -letzter Aufruf zitierter Artikel: 07.05.2018
- [96] WYTZISK, Andreas: *Interoperable Geoinformations- und Simulationsdienste auf Basis internationaler Standards*, Universität Münster, Dissertation, 2003
- [97] X-PLANE: *Internetpräsenz der Flugsimulationssoftware X-Plane des Herstellers Laminar Research*. 2016. – URL <http://www.x-plane.com>. – letzter Zugriff: 07.05.2018
- [98] YOUTUBE: *YouTube ist eine Videoportal des Unternehmens YouTube (Limited Liability Company)*. – URL <http://www.youtube.de>. – Benutzer können auf dem Internetportal Videoclips ansehen, bewerten, kommentieren und selbst hochladen. Eine Qualitätsprüfung oder Rezension erfolgt in der Regel nicht. -letzter Aufruf Videos: 15.04.2017

Abbildungsverzeichnis

1.1	Gebäudeschäden durch einzelne 3D-Gebäudemodelle	3
2.1	Aufbau eines Simulators	8
2.2	Sichtsystem in einem Simulator	10
2.3	Vergleich von realer und synthetischer Umwelt	12
2.4	Ablauf einer verteilten Simulation	14
2.5	Beispiel für eine Joint Exercise	15
2.6	LCIM - Interoperabilitätsstufen	16
2.7	Klassifikation von Waffeneinsätzen	23
2.8	Beispielergebnisse einer Verwundbarkeitssimulation	25
2.9	Level of Detail CityGML	30
2.10	Vergleich SEDRIS und CityGML	32
2.11	Geometrische LoD	33
2.12	Geotypische und geospezifische 3D-Gebäudemodelle	35
2.13	Gesamtarchitektur VIntEL	36
2.14	Vorgehaltene Zerstörungsstufen im SEDRIS-GML-Schema	37
2.15	Prozessmodell der Schadensvisualisierung in VIntEL	39
3.1	Sichtbarkeit nach Detonationen	44
3.2	Gegenüberstellung von virtuellen Stadtmodellen und Fotos der Realwelt	46
3.3	Realphysikalische Erzeugung eines Gebäudes	48
3.4	Beispiele des Voronoi Fracturing	49
3.5	Zeitreihe einer Voronoi-Schadensvisualisierung	50
3.6	Zeitreihe Schadensvisualisierung VBS 3	51
4.1	Prozessmodell zur Erzeugung von Gebäudeschäden	54
4.2	Zeitreihe einer Gebäudesprengung	61
4.3	Anwendung der Schadenstransformation	62
4.4	Erweitertes Prozessmodell der Gebäudeschäden	70
4.5	Voxelisierung	71
4.6	Modifikation	71
4.7	Retriangulierung	72
4.8	Marching Cubes Algorithmus	73
5.1	Schnittbildverfahren	75

5.2	Schnittbilder der Voxelisierung	76
5.3	Testarchitektur des 3D-Impactservice	83
5.4	Gebäudeschäden bei variierender Auflösung der Schadensstufe 2	86
5.5	Gebäudeschäden bei variierender Auflösung der Schadensstufe 3	87
5.6	Gebäudeschäden bei variierender Auflösung der Schadensstufe 4	88
5.7	Gebäudeschaden in X-Plane (1)	93
5.8	Gebäudeschaden in X-Plane (2)	94
5.9	Gebäudeschaden in X-Plane (3)	94
5.10	Gebäudeschaden in X-Plane (4)	95
6.1	Architektur der pragmatischen Interoperabilität	100

Tabellenverzeichnis

1.1	Ziele dieser Arbeit	3
2.1	Übersicht der Simulationsbegriffe	7
2.2	Inhaltliche Übersicht der nachfolgenden Abschnitte.	11
2.3	Geometriemodelle	27
2.4	Vergleich Standards 3D-Gebäudemodelle	28
2.5	Vergleich LoD CityGML	30
2.6	Parameter der Munition Detonation Interaction	38
3.1	Vergleich der Berechnungszeiten nach Ali et al.	48
3.2	Vergleich alternativer Lösungsansätze	52
4.1	Vergleich Parametrisierungen	58
5.1	Laufzeitvergleich der Gebäudeschäden (Schadensstufe 2)	87
5.2	Laufzeitvergleich der Gebäudeschäden (Schadensstufe 3)	88
5.3	Laufzeitvergleich der Gebäudeschäden (Schadensstufe 4)	89
5.4	Vergleich des neuen Ansatzes zu alternativen Lösungsansätze	93