Fraunhofer Institut
Graphische
Datenverarbeitung

Entwicklung eines Konzepts zur Integration von Simulationen ins E-Learning

Vorgelegt von:
Fraunhofer-Institut für Graphische Datenverarbeitung
Institutsteil Rostock
Joachim-Jungius-Straße 11
18059 Rostock

Autoren:
Dipl.-Inf. Mirko Ebert
Alexander Paschen
Telefon: +49 (0) 381 / 40 24 -110
Fax: +49 (0) 381 / 40 24 -199
E-Mail: mirko.ebert@igd-r.fraunhofer.de

Dezember 2007

Berichtsnummer: 07rp011-FIGDR

Inhalt

Abkürzungsverzeichnis

ABK	Abkürzung
ADL	Advanced Distributed Learning
AICC	Aviation Industry Computer Based Training
API	Application Program Interface
ARIADNE	Alliance of Remote Instructional Authoring &Distribution Networks for Europe
CAM	Content Aggregation Model
CBT	Computer Based Training
CMI	Computer Managed Instructions
CMS	Content Management System
DMSO	Defense Modeling and Simulation Office
DoD	Department of Defense
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers
IMS	Instructional Management Systems
JADL	Joint ADL Co-Laboratory
LCMS	Learning Content Management System
LMS	Learning Management System
LOM	Learning Objects Metadata
LTSC	Learning Technology Standards Committee
RTE	Runtime Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Aggregation Model
SISO	Simulation Interoperatbility Standards Organization
SOA	Schools of the Air
WBT	Web Based Training
XML	Extensible Markup Language

1 Einleitung

Lehren und Lernen sind wichtige Grundlagen einer Gesellschaft, die es ständig zu verbessern gilt. Und wie auch in allen anderen Bereichen unseres Daseins, gewinnt hier der Einsatz von Computern mehr und mehr an Bedeutung. Das daraus resultierende E-Learning ist vielleicht nicht jedem ein Begriff, jedoch dürften die meisten unbewusst schon Kontakt mit dem „elektronischen Lernen" gehabt haben. E-Learning hat die Chance, sowohl Qualität als auch Präsenz und Verfügbarkeit von Wissen vermittelnden Inhalten deutlich zu verbessern. Dazu ist es natürlich erforderlich, das E-Learning selbst ständig zu verbessern und seine Möglichkeiten zu erweitern. Hier gibt es z.B. so genannte Learning Management Systeme, welche eine Art personalisiertes Klassenzimmer darstellen, mit individuellen Kursen für den Lernenden, und komplett über das Internet verfügbar sind. Dieses System wird gerade bei Unternehmen zur internen Mitarbeiterschulung immer gefragter. Als Standard hierfür scheint sich das Sharable Content Object Reference Model (SCORM) durchzusetzen. Die Möglichkeiten zur Wissensvermittlung für ein SCORM System sind aber ähnlich begrenzt wie die eines normalen Lehrbuches. Zwar gibt es den Vorteil der individuellen Reihenfolge der dargebotenen Inhalte, jedoch beschränken diese sich trotzdem meist nur auf Texte und Bilder. Geht man aber von der Forschung zur menschlichen Psychologie aus, welche eindeutig besagt, dass unsere kognitive Verarbeitung bei analogen Reizen schneller und leichter funktioniert, und das Lernen aus selbst erfahrenen Ereignissen ebenfalls intensiver erfolgt, so ist es verwunderlich, dass z.B. dem Einsatz von Simulationen kaum Beachtung geschenkt wurde. Denn gerade Simulationen bieten das Lernen durch Betrachten und Erforschen. Natürlich gibt es für viele Anwendungen Simulationen, z.B. Flugsimulationen. Aber hier mangelt es wiederum an lernbegleitenden Ressourcen wie virtuellen Tutoren, oder die Verfolgung einer Lernstrategie durch einen kompletten Unterrichtskurs. Optimal wäre es also, beide Konzepte zu vereinen. Und da SCORM bis jetzt die einzige Referenz für LMS ist, bietet sich dieses als Grundlage für eine Erweiterung von LMS mit Simulationen an. Ziel dieser Arbeit ist es, ein Konzept zur Integration von Simulationen in SCORM zu erarbeiten, und anschließend dieses durch einen Prototypen zu testen. Das heißt, dass Simulationen in Lernkursen für SCORM LMS als zusätzliches Medium nutzbar sein sollen. Dazu werden verschiedene Konzepte, unter anderem von Advanced Distributed Learning (ADL), den Entwicklern von SCORM, untersucht und bewertet werden. Ein wichtiger Faktor wird dabei die Interoperabilität von SCORM LMS spielen. Da SCORM ein Standard ist, würden Veränderungen zur Inkompatibilität mit anderen SCORM LMS führen. Es wird überlegt werden müssen, wie weit dieser Standard verändert werden darf und muss, oder ob sich dies auch vermeiden lässt. Weiterhin sollen Simulationen nicht einfach nur technisch, sondern auch inhaltlich in LMS integriert werden. Die Möglichkeiten, die SCORM einem normalen Lernkurs bietet, sollten auch durch Simulationen nutzbar sein. Das können z.B. zu erreichende Lernobjekte oder zu lösende Aufgaben in einem Simulationslauf sein, oder

dessen Unterbrechen und Fortsetzen.

Diese Arbeit ist in fünf Kapitel gegliedert. Nach dieser Einleitung erfolgt eine Einführung in die Grundlagen des E-Learning, besonders LMS und SCORM. Die Einführung beinhaltet weiterhin einen Überblick über Simulationen und die dieser Arbeit zugrunde liegende Sicht auf diese. Als letztes wird das Thema Simulation im E-Learning näher erläutert. Im Konzeptionsteil werden die besonderen Herausforderung der Integration von Simulationen in SCORM gezeigt und vier Konzepte vorgestellt und diskutiert. Das Kapitel Umsetzung wird eines der Konzepte aufgreifen und eine prototypische Implementierung beinhalten. Als letztes wird das Fazit die Ergebnisse dieser Arbeit zusammenfassen.

2 Einführung

2.1 E-Learning

2.1.1 Begriff

Der Begriff E-Learning leitet sich von „electronic learning" (elektronisch gestütztes Lernen) ab. Damit gemeint ist der Einsatz von modernen Technologien wie z.B. Computer und Internet zur Vermittlung und Verbreitung von Wissen. Das Wort E-Learning lässt viel Interpretationsspielraum zu, weshalb auch keine einheitliche Definition existiert. So kann z.B. der bloß Einsatz von Taschenrechnern oder Computern bei der Textverarbeitung nicht als E-Learning bezeichnet werden. Die meisten Definitionen versuchen deshalb, den Begriff des E-Learning durch verwendete Technologien, Tätigkeiten und Eigenschaften der Lernsysteme zu umschreiben.

2.1.2 Erklärung

Nach einer Definition aus [STANGLE] ist E-Learning „allgemein betrachtet eine besondere Form des computergestützten Lernens, für das charakteristisch ist, dass die genutzten Lernsysteme und -materialien

- In digitalisierter Form dargeboten werden,
- Sich durch Multi- und/oder Hypermedialität auszeichnen,
- Interaktivität zwischen dem Lernenden, dem System, dem Lehrenden und den Mitlernenden unterstützen und
- Online für den Nutzer direkt verfügbar sind."

In dem letzten Punkt geht diese Definition einen Schritt weiter als die meisten anderen Definitionen. Denn die Online-Verfügbarkeit trifft auf sehr viele heutzutage genutzte elektronische Lernmedien nicht zu, wie eine gleich folgende Übersicht zeigen wird. Trotzdem sind diese als E-Learning einzustufen und haben zurzeit auch noch einen hohen Stellenwert. Mit fortschreitender Zahl der Internetzugänge sollte es aber für jedes E-Learning Programm Ziel sein, sein Angebot online verfügbar zu machen.

2.1.3 E-Learning-Systeme

Es gibt heutzutage eine Vielzahl unterschiedlicher E-Learning-Varianten. Sie unterscheiden sich sowohl in den verwendeten Technologien, in ihrer Infrastruktur als auch im didaktischen Szenario und werden je nach Bedarf verwendet. Es ist und wird wahrscheinlich auch nicht möglich sein, **das** E-Learning-System zu entwickeln. Es wird eher auf eine Verbindung verschiedener bestehender Systeme hinauslaufen. Es ist davon auszugehen, dass „Learning Management Systeme“ und „Learning Content Management Systeme“ aufgrund ihrer Flexibilität und Online-Fähigkeit dabei die Basis bilden.

Nichtsdestotrotz spielen andere Systeme und Methoden schon allein wegen ihrer weiten Verbreitung auch in Zukunft eine große Rolle. Da sich diese Arbeit nur auf die technologische Weiterentwicklung von LMS bezieht, werden nur exemplarisch die Systeme, die auf unterschiedlichen Technologien basieren, kurz erklärt. Zu den verschiedenen didaktischen Szenarien sei auf [SD] verwiesen oder andere zahlreiche zu diesem Thema verfügbaren Literaturquellen.

Web- und Computerbasierte Trainingsanwendungen

Computerbasierte Trainingsanwendungen (Computer based Training – CBT) sind Lernprogramme (oder auch Lernsoftware), die meist über CD oder DVD vertrieben werden und lokal auf dem Rechner gespeichert werden. Der Anwender kann diese zeitlich und räumlich flexibel nutzen und muss dabei nicht in direktem Kontakt mit dem Lehrenden stehen. Der Lerninhalt kann über Text, Bild, Ton, Video oder Animation etc. dargestellt werden. CBT können nach Grad der Interaktion unterteilt werden:

- Simulationssysteme, mit denen Sachverhalte spielerisch durch den Lernenden gelöst werden können z.B. Planspiele.
- Tutorielle Systeme, die auf Eingaben des Lernenden reagieren und unterstützend eingreifen.
- Präsentationssysteme, die ein Lernprogramm multimedial in Modulen abspielen. Der Lernende wird durch ein vorgefertigtes Programm geführt z.B. digitale Bücher.

Der Nachteil von CBT besteht in der fehlenden oder nur asynchronen Kommuni-

kation zwischen den Lernenden oder diesen mit ihrem Tutor. Dieser Nachteil kann durch die Nutzung von Intranet oder dem Internet beseitigt werden. Das so genannte „Web based Training“ (WBT) ist eine Weiterentwicklung des CBT. Dabei werden Lernmaterialien nicht über Datenträger, sondern über das Intranet oder Internet meistens bequem über einen Webbrowser abgerufen. Dadurch ergeben sich vielseitige Kommunikationsmöglichkeiten zwischen dem Lehrenden und anderen Lernenden. Häufig sind E-Mail, Chats, Diskussionsforen oder Audio- und Videoübertragungen in dem Lernsystem integriert.

Die Definitionen von CTB und WBT sind sehr allgemein gefasst und treffen auf die meisten heutzutage eingesetzten E-Learning Systeme zu. Im Gegensatz zu z.B. SCORM LMS (siehe Kapitel 2.1.4) sind diese Systeme aber meist nicht standardisiert. Dadurch können z.B. Lehrinhalte nur schwer von einem System auf das andere Übertragen werden. Eine Komposition von Ressourcen verschiedener Systeme ist ebenfalls nicht möglich, oder nur mit individuellem Aufwand für jeden Lernkurs.

Simulation

Simulationen sind Modelle, welche bedeutsame Eigenschaften der Realwelt abzubilden versuchen, um Lernenden durch freies oder gezieltes Experimentieren oder Beobachten Wissen über strukturelle oder funktionale Eigenschaften des Originals zu vermitteln. Diese Definition ist eher anwendungsorientiert, und soll den Nutzen als E-Learning Erfahrung zeigen. Es ist keine treffende formale Definition, wie sie in der Modellierung und Simulation zu finden ist. Deswegen zum Thema Simulation und Simulation im E-Learning im Abschnitt „2.2 Simulation“ bzw. „2.3 Simulation im E-Learning“ mehr.

Videokonferenz / Teleteaching

Diese Entwicklung ergab sich aus frühen Versuchen, weite Entfernungen in dünn besiedelten Gebieten mittels moderner Technologien virtuell zu überbrücken (daher auch der Begriff „distance learning“ – Distanzlernen). Durch Video- und Tonübertragung werden heutzutage virtuelle Klassenräume geschaffen. Diese Technik ist an sich keine neue Form des Lernens, sondern entspricht dem klassischen Präsenzlernens. Die Teilnehmer können direkt über Sprache, Mimik und Gestik kommunizieren. Die Qualität der Kommunikation wird theoretisch nur durch die zur Verfügung stehende Bandbreite eingeschränkt. Eines der ersten großen und erfolgreichen Teleteaching Anstrengungen war das „Schools of the Air“ [SOA], das Kindern im dünn besiedelten australischen Outback mit der im Jahre 1951 gegründeten ersten Funkschule den Schulunterricht ermöglichte. Die Entwicklung des Teleteaching geht sogar soweit, dass ganze Universitäten virtualisiert werden, wie z.B. die „Open Universities Australia“ [OUA]. Dort wird die Anmeldung, Verwaltung und Prüfung über das Internet und Videosysteme durchgeführt.

Learning Management Systeme

Learning Management Systeme machen aus E-Learning Inhalten ein „modulares

Baukastensystem". Sie erlauben es Organisationen, unternehmensweite Lernplattformen aufzubauen, um Wissens- und Kompetenzziele zu definieren, die Lernbedürfnisse von Individuen und Organisationen zu analysieren, online und offline Kurs- und Lernmaterialien auszuwählen und einzukaufen, sowie unternehmensweite Lerninitiativen und -prozesse zu managen. Das System organisiert und verteilt darüber hinaus Lernressourcen wie die Registrierung, die Verfügbarkeit von Klassenzimmern und Dozierenden. Häufig sind ebenfalls e-Commerce Funktionalitäten (z.B. Billing-, Payment Services) integriert.

Die Granularität von Lernmaterialien, aus denen neue Lerneinheiten zusammengestellt werden können, liegt bei einem LMS auf der Ebene eines gesamten Kurses. Es können also nur ganze Kurse ausgewählt oder zusammengefügt werden. Hier liegt größtenteils der Unterschied zu CMS bzw. LCMS.

Nach [HALL] können LMS folgende Funktionen übernehmen:

- Authoring
- Classroom management
- Competency management
- Knowledge management
- Certification or compliance training
- Personalization
- Mentoring
- Chat
- Discussion boards

Content Management Systeme

CMS vereinfachen das Erstellen und das Administrieren von Online-Inhalten wie Texte, Bilder, News, Werbebanner, etc. und werden zumindest für Websites mit hohem Informations- und Aktualitätsgrad wie Online-Zeitungen, Informations-Portale, Firmen-Portale, Intranets, etc. eingesetzt [HAEFELE]. Folgende Eigenschaften zeichnen ein CMS aus:

- Strikte Trennung von Inhalt und Darstellung
- Komponenten Management, die erstellten Inhalte werden zum besseren Auffinden und Wiederverwenden mit Metadaten versehen und in einer Komponentendatenbank abgespeichert.
- Workflow Management, in einem CMS können Regeln für ein Workflow festgelegt werden, z.B. ob neu erstellte Inhalte von anderen Personen überprüft werden müssen, bevor diese freigegeben werden, oder wie lange Artikel als aktuell eingestuft werden.

Learning Content Management Systeme

Ein Learning Content Management System ist eine Software, die die Erstellung, Speicherung und Verwaltung von wieder verwendbaren Lernobjekten sowie die Organisation und Betreuung webunterstützten Lernens ermöglicht.

LCMS vereinen die typischen Funktionen von Learning Management Systemen mit den Funktionen zur Content-Erstellung und Content-Personalisierung der CMS [HAEFELE].

Hypervideo

Hypervideo ist ein kollaboratives Annotationssystem für Videos, das durch seine Möglichkeiten gut in das E-Learning Konzept passt. Wichtige Elemente im Video können markiert und mit Metainformationen versehen werden. Diese Markierungen werden beim Abspielen mit eingeblendet und versorgen den Nutzer mit zusätzlichen Informationen. Dadurch entsteht gegenüber dem einfachen Lehrvideo ein Mehrwert für den Lernenden. Die Annotationen können meistens auch von den Nutzern selbst erstellt oder erweitert werden. In kollaborativen Lerngruppen können so z.B. Fragen direkt zu einem sichtbaren Element im Video gestellt und beantwortet werden.

2.1.4 Standards

Überblick

Obwohl die Entwicklung des E-Learning schnell voranschreitet und es ein vielfältiges Angebot gibt, wird man sehr bald merken, dass man sich immer nur einem begrenzten Teil widmen kann, da die meisten Systeme inkompatibel zueinander sind. Wenn man ein z.B. ein System für Online-Lehrkurse besitzt, und dieses mit Kursen von anderen Anbietern erweitern möchte, ist dies meist nicht möglich, ohne Anpassungen am Kurs oder der kompletten Neuerstellung. Dieses Problem kann durch Standards vermieden werden.

Ziele der Standardisierung sind zum einen das "Content Sharing" (vergl. Projekt "Content Sharing" des Fraunhofer IGD Rostock), das den einfachen Austausch von Kursmaterial zwischen verschiedenen E-Learning Systemen erlaubt. Dazu müssen die digitalen Lerneinheiten durch ein einheitliches Metadatenformat beschrieben sein und die Lerninhalte in fest definierten Strukturen vorliegen. Die Metadaten übernehmen dabei die Rolle der Katalogisierung ähnlich in einer Bücherei. Einheitliche Strukturen ermöglichen den korrekten Zugriff auf die Inhalte. Außer der Standardisierung der eigentlichen Lernressourcen gibt es auch Bestrebungen, ganze Lernsysteme zu definieren, also die Interaktion zwischen Lernendem, Kurs und Lernsystem. Dies hat besonders bei LMS und CMS große Bedeutung.

Die folgende Tabelle zeigt die zurzeit am weitesten verbreiteten E-Learning Stan-

dards und Gremien.

Standard/Gremium	Kurzbeschreibung
Dublin Core	Satz von Metadaten-Spezifikation, soll die Suche nach digitalen Dokumenten und das Wiederauffinden von elektronischen Ressourcen erleichtern (kein reiner E-Learning Standard).
Ariadne	Projekt der Europäischen Kommission, Empfehlungen für Educational Metadata enthält Spezifikationen mit mehreren Kategorien.
IMS	Initiative IMS (Instructional Management Systems), weitreichende Spezifikationen auf der Basis von XML.
IEEE Learning Objects Metadata (LOM)	Beschreiben Standards für die Syntax und Semantik der Metadaten für Lernobjekte. Metadaten werden definiert als Attribute, die ein Lernobjekt angemessen und vollständig beschreiben.
AICC	Weit verbreiteter Standard zur Entwicklung von CBTs, Standards aus der Luftfahrtindustrie.
SCORM	Von ADL entwickelte Referenzspezifikation für LMS. Vereinigt einige Standards von Ariadne, IMS, IEEE LTSC und AICC zu einem Referenzmodel für LMS

Bis jetzt ist SCORM der einzige relativ weit verbreitete Standard für LMS und E-Learning-Inhalte.

Metadaten

Metadaten sollen das E-Learning Leben leichter machen. „Metadaten sind Informationen über Ressourcen/Objekte mit dem Ziel, Wissensmodule so zu charakterisieren, dass sie im jeweiligen Kontext treffsicher gefunden werden." [Meder] Idealerweise könnten so alle weltweit verfügbaren Wissensressourcen über ein einheitliches Suchsystem gefunden und genutzt werden. Und nicht nur das Finden, sondern auch das Vergleichen spielt eine Rolle. Kurse mit ähnlichem Inhalt können z.B. unterschiedlich didaktisch aufbereitet sein, andere Lernstrategien verfolgen, verschiedene Medien verwenden (mehr oder weniger modern) oder das Wissen durch bloßes darbieten oder projektbezogenes Arbeiten vermitteln. Durch gute Metadatenmodelle und deren konsequenten Einsatz könnten so für den Nutzer optimal zugeschnittene Kurse ausgewählt werden. Weitere Vorteile sind die Wiederverwendung bereits erstellter Lerneinheiten in anderen Kontexten, Erstellung individueller Lernpfade durch Zusammenstellung verschiedener Lerneinheiten und die automatische Anpassung des Lernpfades an den Lernfortschritt des Nutzers. Weitere Untersuchungen, wie Metadaten in E-Learning Umgebungen zum Einsatz kommen könnten, lassen sich in der Diplomarbeit [CAMPEN] nachlesen.

Im praktischen Einsatz findet die Verwendung von Metadatenstandards allerdings kaum Anklang laut [Meder]. Überhaupt werden Metadaten kaum genutzt.

Dadurch sind auf diese Weise angebotene Lerninhalte immer nur für einen kleinen Personenkreis nutzbar. Das zweite Problem ist ein fehlender einheitlicher Standard für Metadaten. Es gibt zwar Versuche, die Beschreibung von Metadaten zu standardisieren (Learning Objects Metadata, Dublin Core, Instructional Management Systems Project). Allerdings konnte sich nach [Meder] bis jetzt keines der dreien durchsetzen. Als Gründe nennt er dafür die zu langsame Integration der immer weiterentwickelten Technologien des Internets und fehlende Akzeptanz. Bei Akzeptanz ist nicht nur der technische Aspekt gemeint, sondern auch der Willen, überhaupt Lernmodule von fremden Autoren in das eigene Lehrangebot zu integrieren. Bei dem Versuch, "Wissenspools" als Basis für Universitäten oder andere Bildungseinrichtungen, die schon digitale Lernmodule im Unterricht verwenden, einzurichten, hat sich gezeigt, dass nur in Einzelfällen fremde Lernmodule zur Wiederverwendung herangezogen wurden. Die Umsetzung des Wissenspools belief sich meist auch nur auf eine Linksammlung und Anhäufung von unterschiedlichsten Materialien. Dabei lag das Problem nicht etwa in der technischen Durchführbarkeit oder an unzureichenden finanziellen Mitteln, sondern an sozialen und psychlogischen Aspekten wie Konkurrenzdenken und die Scheu vor dem Fremden. Dieses Beispiel soll zeigen, dass bei der Entwicklung und Durchsetzung auch der Faktor Mensch eine entscheidende Rolle spielt.

Die drei relevanten Standards für Metadaten sind:

- **Dublin Core (DC): 15 Metadaten Elemente**
- title, creator, subject, description, publisher, contributor, rights
- einfach, überschaubar, flexibel, erweiterbar
- **Draft Standard for Learning Objects Metadata (LOM)**
- General (title, description, keywords)
- Lifecycle (development, current state)
- Meta-Metadata
- Technical (media, type, size, software requirements)
- Educational (pedagogical characteristics: interactivity level, typical learning time, semantic density)
- **Instructional Management System (IMS)**
- Ziele: Suche, Management, Wiederverwendbarkeit pädagogischer Ressourcen
- De facto Standard
- Erfasst Lernziele (Educational Objective), Lernvoraussetzungen (Prerequisite), Kommentare zur pädagogischen Methode (Description)
- Beschreibung inhaltlich zusammenhängender Module möglich (komplexere

Lerneinheiten)

SCORM (Shareable Content Object Reference Model)

Überblick

Das Shareable Content Object Reference Model wurde von der Advanced Distributed Learning Initiative (ADL) entwickelt. Es vereinigt frühere Bestrebungen der Standardisierung im E-Learning der Organisationen IMS, AICC, IEEE und Ariadne zu einem Referenzmodel für wieder benutzbaren E-Learning Content. Die aktuelle Version ist SCORM 2004. Da das in dieser Arbeit verwendete LMS noch die vorherige Version SCORM 1.2 verwendet, werden sich die genaueren Ausführungen darauf beschränken.

SCORM setzt sich aus so genannten „Books" zusammen. In der Version 1.2 sind das „Content Aggregation Model" (CAM) und „Runtime Environment" (RTE). Die neuere 2004er Version enthält zusätzlich „Sequencing and Navigation". Die Aufgabe von CAM ist es, das Zusammenstellen von Lerninhalten aus wieder auffindbaren, wieder verwendbaren, austauschbaren und interoperablen Quellen zu ermöglichen. RTE definiert die Schnittstelle zwischen SCORM konformen Lerninhalten und LMS. Es gibt Compliance Suits, die die Lerneinheiten oder ein LMS auf SCORM Konformität testen. Dabei ist ein konformes LMS allerdings nicht zwangsläufig benutzbar. Es wird lediglich die Bereitstellung bestimmter API Funktionen überprüft.

Content Aggregation Model (CAM)

Das Content Aggregation Model definiert Struktur, Inhalt und Verhalten von Lerninhalten von den Basisbausteinen bis zum komplexen Lernmodul. CAM ist noch einmal unterteilt in die 3 Bereiche Content Model, Meta-Data und Content Packaging.

Content Model

definiert die eigentlichen Lerninhalte - den Zusammenhang zwischen Assets, Sharable Content Objects (SCO) und Content Aggregation. Basis sind die minimalen elektronischen Lerneinheiten, sogenannte Assets. Das können Text, Bilder, Audio, Video, Webseiten usw. sein, alle Formen von Daten, die über einen Webclient geliefert werden können.

SCO

Ein SCO repräsentiert eine Ansammlung von mehreren Assets, wovon ein spezielles beim Ausführen die Kommunikation mit dem LMS über die Runtime Environment übernimmt. Ein SCO ist die kleinste durch die RTE auffindbare Lerneinheit. Es sollte unabhängig von anderen SCO verwendet werden können und in seiner logischen Struktur so aufgebaut sein, dass es die kleinste wieder verwendbare, auffindbare und austauschbare Lerneinheit ist, und dabei ein in sich geschlossenes minimales Lernziel erreicht. Dadurch können SCO zu immer neuen individuellen Lernmodulen zusammengesetzt werden.

Die minimalen Anforderungen an ein SCO sind, dass es den API Adapter der RTE sucht, um sich mit der RTE zu verbinden, und die API Funktionen *LMSInitialize()* und *LMSFinish()* aufruft. Ein SCO muss von jedem SCORM konformen LMS in der gleichen Weise gestartet werden können. SCOs dürfen selbst keine anderen SCOs starten.

Content Aggregation

gibt den einzelnen Lernressourcen in einem Packet eine Ordnung, wie Kurs, Kapitel, Modul usw. und legt die Reihenfolge fest, in der die Ressourcen dem Lernenden präsentiert werden sollen (Content Structure). Es ist Aufgabe des LMS, dieses zu interpretieren und die Reihenfolge einzuhalten. Fest verlinkte Lerneinheiten werden vom LMS als eine ganze Einheit angesehen und können nicht einzeln angesteuert werden. Das Referenzieren von externen Lernressourcen sollte auf jeden Fall vermieden werden, da beim Austausch von Kursen zwischen den LMS die externe Ressource nicht verfügbar sein kann.

SCORM Metadata

beschreibt, wie die Standards von IEEE und IMS auf die einzelnen CAM Komponenten angewendet werden. Es gibt Vorgaben für das Erstellen von Metadaten für Assets, SCOs und Content Aggregation, um diese in einer konsistenten Art und Weise zum Zwecke des Auffindens und der Wiederverwendung zu beschreiben. Es gibt 3 Teilbereiche für das SCORM Metadaten Model:

Information Model

Beschreibt die Hierarchie und Verwendung der Metadaten Elemente. Das Infor-

mation Model ist dasselbe wie IEEEs LOM Spezifikation. Die Elemente werden in 9 Kategorien eingeordnet. Für jedes gibt es Angaben zu Position in der Hierarchie, Name, Verwendung, Häufigkeit und Datentyp. Die Codierung wird hier offen gelassen.

SCORM Meta-data XML Binding

Inhalt und Codierung der Metadaten sind getrennt. Das Format der Metadatenelemente wird zurzeit durch die „Learning Ressource Meta-Data XML Specifikation" von IMS definiert, welches, wie der Name schon sagt, XML als Grundlage für die Codierung verwendet.

SCORM Meta-data Application Profiles

Definiert, welche Metadatenelemente wie benutzt werden auf Anwendungsebene für Assets, SCO und Content Aggregation, um SCORM konformen Inhalt zu erstellen.

Content Packaging

Zum Erstellen komplexer Kurse werden SCOs und Assets zu Packeten zusammengefasst. Content Packaging beschreibt diesen Vorgang. Wichtig dabei sind Struktur der Dateien, Organisation der Lerneinheiten, wie ein gewünschtes Verhalten der Lernheiten erzielt wird, und wie Packete zwischen verschiedenen Systemen ausgetauscht werden können. All diese Informationen werden in einer Manifestdatei gespeichert, welches Basis jedes SCORM Kurses ist. Als Standard wird dazu die IMS „Content Packaging Specification" benutzt.

Runtime Environment

Die Runtime Environment ist der Teil von SCORM, der von einem LMS implementiert werden muss, um SCORM konform zu sein. Diese setzt sich zusammen aus einer Application Program Interface (API), Datenmodell und Launch. Die API dient als Kommunikationsbrücke zwischen Lernressourcen und LMS. Mit ihr können Statusmeldungen und Daten ausgetauscht werden. Das Datenmodell bestimmt die Art der Informationen, die über die API kommuniziert werden. Eine Erweiterung des Datenmodells ist möglich, allerdings schränkt das die Interope-

rabilität ein. Launch legt fest, wie die Kommunikation zwischen LMS und Lernressource starten muss, um eine Verbindung herzustellen.

Launch

Es gibt einige Regeln für das Starten von Lernressourcen. Nur SCOs können mit dem LMS kommunizieren. Nur ein SCO darf gleichzeitig mit dem LMS kommunizieren. SCOs dürfen keine anderen SCOs starten. Die Reihenfolge, in der SCOs oder Assets aufgerufen werden, wird durch das LMS bestimmt. Richtlinien für „Sequencing and Navigation" werden erst mit SCORM 2004 eingeführt.

API

Die API basiert direkt auf der AICC API Spezifikation CMI001 aus dem Dokument „Guidelines for Interoperability". Diese liefert Methoden zum Zustandswechsel, Fehlerbehandlung und Datenaustausch zwischen SCO und LMS. Die Kommunikation geht dabei immer nur vom SCO aus, auf keinen Fall anders herum. Um die API einem SCO zur Verfügung zu stellen, muss das LMS einen API Adapter bereitstellen. Unterstützt werden dafür bis jetzt nur Javascript Funktionsaufrufe.

Data Model

Das verwendete Datenmodell basiert ebenfalls auf AICCs CMI Data Model, beschrieben im „Guidelines for Interoperability" Dokument. Zukünftige Versionen von SCORM werden wahrscheinlich auch weitere Modelle unterstützen. Das ursprüngliche CMI Modell wurde reduziert, da viele Elemente nicht von Entwicklern von LMS oder Lerninhalten genutzt worden sind. Die jetzt verwendeten sind hierarchisch durch ihren Namen geordnet. Z.B. „cmi.core.student_id" gibt an, dass es sich um das CMI Modell handelt und das Datenfeld Studenten-Id in der Rubrik „core" verwendet wird. Generell müssen alle Elemente aus Core von einem LMS implementiert werden, um minimal SCORM konform zu sein. Weitere Rubriken sind optional. Das heißt für SCOs, dass das fehlen dieser Elemente mit berücksichtigt werden muss.

2.1.5 Das LCMS smartBLU

Das am Fraunhofer IGD Rostock entwickelte System smartBLU ist ein Java-basiertes LCMS, das dem SCORM Standard 1.2 entspricht. Die in dieser Arbeit zu ent-

wickelnde Erweiterung für Simulationen soll für dieses System erfolgen.

2.2 Simulation

2.2.1 Begriff

Zunächst soll geklärt werden, wie der Begriff Simulation in dieser Arbeit zu verstehen ist. Heutzutage wird dieses Wort sehr leichtfertig für alles verwendet, was etwas mit der Ausführung von oder Tätigkeit am Modell statt in der Realität zu tun hat. Im Allgemeinen geht es bei der Betrachtung von Simulationen um die Unterscheidung der verschiedenen mathematischen Modelle und deren formalen Eigenschaften sowie die effiziente Ausführung der Modelle. Dies soll hier als akademische Sicht bezeichnet werden. Für diese Arbeit wird allerdings eine andere Betrachtungsweise benötigt, eine, die auf die Nutzungseigenschaften eingeht. Dies soll Anwendungssicht genannt werden. Beiden Sichten liegt natürlich die gleiche Definition für Simulationen zugrunde. Die akademische Sicht ist deshalb ungeeignet, da es hier um die Frage geht, wie Simulationen zum Zwecke des Lernens verwendet werden können, genauer, wie Simulationen mit einem LMS wie smartBLU verknüpft werden können. Dabei ist es für den Lernenden unerheblich, nach welchem Modell oder Formalismus eine Simulation arbeitet. Allein der wiedergegebene Inhalt zählt. Die Konzeption im Abschnitt 3 wird sich daher nur auf die Anwendungssicht stützen.

Zum besseren Verständnis werden die akademischen Aspekte kurz erwähnt mit Verweis auf ausführlichere Quellen. Die für die hier gestellte Aufgabe wichtigen Unterscheidungskriterien einer Simulationen erfolgen aber im Abschnitt 2.2.5.

2.2.2 Definition

Im Abschnitt 2.1.4 „E-Learning Systeme“ viel der Begriff Simulation bereits mit einer aus E-Learning-Sicht motivierten Erklärung. Die folgenden zwei Zitate geben diese aus einer etwas wissenschaftlicheren Sicht wieder:

> *”A simulation is an experiment performed on a model” (Korn/Wait, 1978) [KW78]*
>
> *”Modeling means the process of organizing knowledge about a given system” (Zeigler, 1984) [ZEI84]*

Grundlegend wird zwischen Simulation und Modell unterschieden. Das Modell beherbergt alle Eigenschaften und Gesetze der zu simulierenden Objekte. Diese können aus der realen Welt abstrahiert werden oder auf fiktive Konstrukte aufbauen. Eine Simulation ist dann die Ausführung des Modells.

2.2.3 Nutzen und Ziele

Im Allgemeinen werden folgende Ziele und Motivationen für die Modellierung und Simulation genannt

- wenn ein Problem analytisch nicht lösbar ist
- oder Experimente am realen System nicht möglich oder sinnvoll sind
- zu kostspielig
- zu zeitaufwendig
- Beobachtung zu schwierig
- zu riskant
- Experiment irreversibel (Ausgangszustand kann im System nicht wiederhergestellt werden)
- Fragestellungen
- Prüfen von Hypothesen
- Vorhersage
- Verbesserung des realen Systems
- Entwurf des realen Systems
- Lehre
- Unterhaltung

Die meisten Anwendungen von Simulationen entstehen aus den ersten vier genannten Punkten unter Fragestellungen. Die letzten beiden Punkte, Lehre und Unterhaltung, wurden bis jetzt meist außer Acht gelassen und anderen Bereichen überlassen. Seit das Thema E-Learning sich auf viele Wissenschaftsbereiche ausbreitet, gibt es auch im Bereich Simulation Forschungen in Richtung Trainingsanwendungen und Spiele. Hier ist man sich allerdings nicht einig, inwieweit Spiele oder Trainingsszenarien bereits selbst schon Simulationen sind (Simulationen werden im Bereich E-Learning auch nicht immer als dazugehörig angesehen). Aus Sicht des Anwenders ist dies zutreffend. Man denke nur an die Fülle an Renn-, Wirtschafts- oder Flugsimulationsspielen. Diese sind gleichzeitig Spiele und Simulationen, haben aber nicht den formalen Charakter eines Simulationssystems (Trennung Modell / Simulation, Anspruch auf Validität des Modells usw.) und dienen meist auch nicht als Informationsquelle für reale Systeme. Ähnlich verhält es sich bei virtuellen Trainingsszenarien wie Brandbekämpfungsübungen, Wiederbelebungsmaßnahmen oder Kriegssimulationen beim Militär, wobei die Validität des Modells bei solchen Anwendungen schon wieder an Wichtigkeit zunimmt.

Um das breite Spektrum an Anwendungsmöglichkeiten für Simulationen zu zei-

gen, sei hier eine Liste typischer Bereiche genannt:

- Spielsimulationen - z.B. Flugsimulationen, Rennsimulationen, Wirtschaftssimulationen
- Unternehmenssimulation für die Aus- und Weiterbildung - z. B. Unternehmensplanspiel, Business Games
- Meteorologische Simulation zur Wettervorhersage
- Physikalische Simulation und astrophysikalische Simulation
- Chemische Simulation
- Biologische Simulationen, z. B. Simulation neuronaler Netze
- Sozioökonomische Simulation, z. B. Multiagentensysteme
- FEM (Finite Elemente Methode)

2.2.4 Formale Einteilung

Hier soll ein kurzer Überblick über die formale Einteilung von Simulationen, wie sie meist in gelehrt wird, erfolgen. Dies Informationen sind eher für die Erstellung von Modellen oder den Entwurf von Simulationssystemen wichtig, was hier natürlich nicht weiter ausgeführt werden kann. Eine Schnittstelle zu einer Simulation sollte auch immer so abstrakt sein, dass Kenntnisse über die internen Vorgänge nicht nötig sind.

In der Simulation werden meistens folgende Gebiete genannt:

- **Monte Carlo Simulation** oder stochastische Simulation (nach John von Neumann und Stanislaw Ulam)
- "a situation in which a difficult non-probabilistic problem is solved through the invention of a stochastic process that statisfies the relations of the deterministic problem" (Morgenthaler, 1961) [MOR61]
- **Kontinuierliche Simulation**
- Gleichungsmodelle, in jedem Zeitintervall unendliche viele Veränderungen, explizite Beziehungen zwischen Zustand und Zeit und den daraus resultierenden Diskontinuitäten sind von Interesse
- **Diskrete Simulation**
- in jedem Zeitintervall sind nur endlich viele Veränderungen von Interesse (d.h. Zustandsänderungen zu präzisen Zeitpunkten)
- **Hybride Simulation**
- Hybride Simulationen verbinden Verfahren zur Diskreten und Kontinuierlichen Simulation. Die Simulation verläuft von diskretem Ereignis zu diskretem

Ereignis kontinuierlich.

- Numerische Simulation
- Robotik
- Multi-Agenten-Simulation
- Makro- und Mikrosimulation

Modelle

Modelle stellen eine Abstraktion des zu simulierenden Systems dar (Struktur, Funktion, Verhalten), das heißt, es werden gewisse Eigenschaften den Systems vereinfacht. Die Eigenschaften müssen so gewählt und abgebildet werden, dass die aus den Experimenten mit der Simulation gewünschten Ergebnisse möglichst mit dem wirklichen System

übereinstimmen. Dieser Punkt spielt deswegen eine wichtige Rolle, da sich aus einem invaliden Modell keine brauchbaren Aussagen ableiten lassen. Das trifft sowohl für Experimente, als auch für Simulationen mit Lehrzwecken zu. Es wäre sicherlich fatal, wenn ein Flugsimulator für Pilotentraining sich grob anders verhält als das reale Flugzeug.

Ein Modell kann aus verschiedenen Quellen gewonnen werden. Z.B. empirisch durch Beobachtungen und Vermutungen über das System, durch Ableitungen von einer Theorie oder spontanes „Erfinden".

Je nach Anwendung und Anforderung an die Simulation werden verschiedene Modelle benutzt. Die Wahl des Modells bestimmt gleichzeitig die Art des Simulators:

- kontinuierliches Modell
- Zellulärer Automat
- DEVS (Discrete Event Specified Systems)
- Warteschlangenmodell
- Mehrebenenmodell
- Multiagentenmodell
- Mikromodell

Ausführlichere Informationen zur Simulation und Modellierung findet man in der unter [MOSI] gelisteten Literaturliste.

2.2.5 Einteilung aus Anwendersicht

Für die spätere Konzeption der Verknüpfung von Simulationen mit einem SCORM konformen LMS ist es nicht unbedingt notwendig zu wissen, nach welchem Formalismus diese aufgebaut ist, sondern wie die Interaktion mit der Si-

mulation funktioniert. Danach entscheidet sich, wie die Simulation zu Lernzwecken eingesetzt werden kann, wie ein Interface aussehen muss und welche Anpassungen am LMS oder der Simulation nötig sind.

Für eine Klassifikation von Simulationen aus der Perspektive der Interaktivität sind andere Kriterien notwendig als für eine formale Einteilung. Da zu diesem Thema bisher kaum Forschung betrieben wurde, gibt es keine anerkannten und evaluierten Konzepte für eine solche Einteilung. Die folgende Übersicht enthält daher die nach meiner Erfahrung ausschlaggebenden Kriterien, anhand derer sich Simulationen aus Anwendungssicht beschreiben lassen können:

1. Interaktivität
    1. Kann der Nutzer die Simulation nach dem Start steuern?
    2. Erfolgt die Steuerung über ein grafisches Interface oder Kommandozeile?
2. einmalige / schrittweise Berechnung und Dauer
    1. Wird das Ergebnis einmalig berechnet und dann präsentiert?
    2. Wie lange dauert die Berechnung?
    3. Ist es eine fortlaufende Simulation, deren aktueller Fortschritt beobachtet werden soll?
    4. Kann der Fortschritt gespeichert und später fortgesetzt werden?
    5. Gibt es ein Ende / eine Abbruchbedingung?
3. synchron / asynchron
    1. Erfolgen Interaktion, Ergebnispräsentation und Visualisierung synchron zur Laufzeit oder asynchron? (Echtzeit)
    2. Gibt es Zeitlimits zur Aufrechterhaltung der Synchronität?
    3. Wie wird eine Session bei asynchroner Simulationen wiedergefunden / wiederaufgenommen?
4. Multi-User
    1. Können mehrere Nutzer gleichzeitig an einer Simulation teilnehmen?
    2. Wie sieht die Teilnahme aus? Gemeinsam interaktiv oder passiv zuschauen?
    3. Gibt es eine Mindest- / Maximalteilnehmerzahl?
5. Visualisierung
    1. Gibt es eine fest Definierte Visualisierung der Ergebnisse oder werden lediglich die Daten als Ergebnis zurückgegeben?
    2. Sind Eingabe und Ergebnisinterface miteinander gekoppelt? (z.B.

Feuerlöscher zur Brandbekämpfung in einer interaktiven 3D Umgebung steuern)

3. Kann die Visualisierungskomponente über Netzwerke geladen werden?
4. Sind die Interfaces Web-Browser-fähig?

Weiter Eigenschaften:

1. Aufzeichnung der Interaktionen
    1. Kann das Verhalten des Nutzers durch den Lehrenden / Lernenden im Nachhinein analysiert werden?
2. Manipulation des Szenarios zur Ausführungszeit durch z.B. Schwierigkeitsgrad
    1. Kann die Simulation dem Verhalten und den Fähigkeiten des Nutzers angepasst werden?

Ausprägung der Kriterien

1. Interaktiv : ja / nein
    1. Wenn ja
        1. Schnittstelle : Grafisches Interface / Texteingabe bzw. Konsole
2. Art und Dauer der Berechnung : einmalig / fortlaufend
    1. Wenn einmalig
        1. Dauer : Sofortiges Ergebnis / bestimmte Zeitdauer / Abbruchbedingung / unbestimmte Zeitdauer
    2. Wenn fortlaufend
        1. Dauer : unbegrenzt / Zeitlimit / Abbruchbedingung (außer Zeitvorgabe)
        2. Wiedereintritt : ja / nein
3. Ausführung : synchron / asynchron
4. Multi-User : ja / nein
    1. Wenn ja
        1. Art : aktiv / passiv
5. Visualisierung : ja / nein
    1. Wenn ja
        1. User Input : separat / integriert
        2. Lokalität : Installation auf dem Client / kann über Netzwerk benutzt

3. werden
4. Wenn über Netzwerk
    1. Webbrowser-fähig : ja / nein

6. Aufzeichnung der Interaktion : ja / nein
7. Manipulation des Szenarios : ja / nein
    1. Wenn ja
        1. Art: adaptiv / manuell

Klassifikation von Simulationen nach den Kriterien

Eine klare Trennung in Klassen von Simulationen, wie es bei der formalen Einteilung geschieht, ist nach diesen Kriterien nicht möglich. Eine Simulation kann entweder diskret, kontinuierlich, hybrid usw. sein (siehe Gebiete der Simulation). Zu diesen verschiedenen Ansätzen gibt es eine Reihe von etablierten Modellen. Aus Anwendungssicht kann man nahezu alle von mir genannten Kriterien beliebig kombinieren, was es schwierig macht, klare Unterscheidungsmerkmale zu finden. Einige Eigenschaften treten aber zwangsläufig immer in der gleichen Konstellation auf. So wird z.B. eine interaktive Simulation immer mit einer fortlaufenden Berechnung einhergehen, da eine Interaktion sonst ohne Auswirkung auf das Ergebnis wäre. Genauso ist die Verwendung von grafischen Benutzerschnittstellen (GUI) zur Interaktion auch nur bei synchroner Präsentation der Auswirkungen auf das simulierte System sinnvoll. Niemand würde Objekte in einer 3D-Umgebung bewegen, wenn man die Rückmeldung davon erst eine Stunde später per E-Mail erhält.

Zunächst sollen Simulationen abseits ihrer Visualisierungsformen, Nutzerinterfaces oder Portierbarkeiten klassifiziert werden, um später eine Konzeption zu erarbeiten, ohne sofort auf Implementierungsprobleme eingehen zu müssen wie z.B.:

- Wie kann man eine Simulation, die eine lokale Installation verlangt, über Web- Browser dem Nutzer zugänglich machen? Ist das überhaupt möglich?
- Die Simulation verwendet zur Visualisierung eine Grafik-Engine. Kann die Ausgabe über Web Browser angezeigt werden?

Aus Meiner Sicht ergeben sich drei Typen von Simulationen: nicht interaktive, interaktive und Multi-User interaktive. Diese drei Varianten werden zur Konzeption im Abschnitt 3 herangezogen.

Typ 1a:

- Interaktiv nein
- Art der Berechnung einmalig

- Dauer sofort / bestimmte Zeitdauer
- Ausführung asynchron
- Multi-User nein

Typ 1b:

- Interaktiv nein
- Art der Berechnung fortlaufend
- Dauer unbegrenzt / Zeitlimit / Abbruchbedingung
- Ausführung synchron
- Multi-User nein

Die Typen 1 haben die wenigste Komplexität in Bezug auf Interaktivität. Sie werden lediglich mit Parametern gestartet und laufen dann selbständig ab. Typ 1a berechnet dabei nur ein Ergebnis, welches entweder sofort bereitgestellt wird oder erst nach einem gewissen Zeitraum, falls z.B. die Approximation des Ergebnisses sehr rechenintensiv ist. Vermutlich findet Typ 1a kaum Verwendung, zumindest nicht als Lernsystem, da es bei Simulationen eher darum geht, Systeme über einen Zeitraum hinweg zu beobachten. Bei Typ1b dagegen kann der Verlauf der Simulation mitverfolgt werden. Der aktuelle Zustand wird meistens auch sofort angezeigt. Bei aufwendigeren Berechnungen kann es bei synchroner Ausführung allerdings längere Zeit dauern, bis ein neuer Zustand eintritt. Hier gibt es vielleicht die Möglichkeit, den Simulationsverlauf zur späteren Wiedergabe aufzuzeichnen. Dann würde ich die Ausführung aber asynchron einstufen. Ein Beispiel für Typ 1b wäre eine Räuber-Beute-Simulation, in der sich die Population von Jäger und Gejagtem über einen längeren Zeitraum entwickelt. Entscheidend sind dabei Startparameter wie Fortpflanzungshäufigkeit, Anfangspopulation oder Fresshäufigkeit anders entwickeln.

Typ 2:

- Interaktiv ja
- Art der Berechnung fortlaufend
- Dauer unbegrenzt / Zeitlimit / Abbruchbedingung
- Ausführung synchron
- Multi-User nein

Typ 2 kann als eine Erweiterung von Typ 1b verstanden werden, wenn die Parameter während der Simulation verändert werden können. Als Parameter sollen dabei alle veränderlichen Faktoren verstanden werden, die Einfluss auf den Simulationsverlauf haben. Das sind sowohl Startparameter, als auch Eigenschaften von Objekten in der Simulation. Ein Typ 2 Beispiel ist eine Simulation für das Löschen von Waldbränden. Hier gibt es sowohl Startparameter wie Temperatur,

Windgeschwindigkeit und -richtung, Luftfeuchtigkeit, als auch durch den Nutzer ausgeführte Aktionen, die den Verlauf verändern. Wo werden welche Brandbekämpfungsmaßnahmen wann durchgeführt. Da dieses Szenario nur durch eine Person durchgeführt wird, dient es eher der strategischen Planung.

Typ 3:

- Interaktiv nein
- Art der Berechnung fortlaufend
- Dauer unbegrenzt / Zeitlimit / Abbruchbedingung
- Ausführung synchron
- Multi-User ja
- Art passiv

Typ 3b:

- Interaktiv ja
- Art der Berechnung fortlaufend
- Dauer unbegrenzt / Zeitlimit / Abbruchbedingung
- Ausführung synchron
- Multi-User ja
- Art aktiv

Typ 3 bezeichnet die Multi-User Simulationen. In Typ 3a haben die Nutzer nur eine passive Rolle, das heißt sie können den Simulationsverlauf lediglich mitverfolgen. Man könnte es eine Art „Public Viewing" nennen, wo die Nutzer eventuell miteinander kommunizieren, aber nicht mit der eigentlichen Simulation interagieren. In Typ 3b geht es um die gemeinsame Interaktionen der Nutzer im simulierten System. Hier bieten sich viele Szenarien an, in denen das sinnvoll ist: Wirtschaftssimulationen, Brandbekämpfung, militärische Übungen usw..

Diese Einteilung von Simulationen wurde hier so gewählt, dass die Komplexität einer Integration in ein LMS bzw. LCMS zunimmt, um im Konzeptionsteil verschiedene Ansätze auf ihre Tauglichkeit für unterschiedliche Simulationssysteme zu überdenken. Offensichtlich nimmt der Integrationsaufwand von Typ1 bis Typ3 zu.

2.3 Simulation im E-Learning

2.3.1 Begriff

Wie in den Abschnitten „2.1 E-Learning" und „2.2 Simulation" schon erläutert, kann man Simulationen selbst schon als E-Learning bezeichnen, da sie der Defi-

nition von E-Learning entsprechen. Allerdings sind die meisten Simulationen nicht als Lehrinstrument entwickelt worden. Bei Simulationen im E-Learning sollte dieser Aspekt mehr berücksichtigt werden. Es geht um die Frage, wie man Simulationen in E-Learning Systemen sinnvoll zur Lehre einsetzen kann. Didaktik und Lernstrategien spielen hier sicherlich eine größere Rolle als optimale Modelle und Effizienz einer Simulation.

Simulationen können in E-Learning Umgebungen allerdings auch andere Aufgaben übernehmen als den eigentlichen Lehrinhalt zu vermitteln. So gibt es beispielsweise auch „Intelligente Tutoring Systeme" (ITS). Hier übernimmt die Simulation die Steuerung und den Ablauf von Lernmaterial. Das ITS versucht zu erkennen, wie gut das Wissen eines Anwenders ausgeprägt ist, und passt entsprechend dieser Ausprägung den zu vermittelnden Inhalt an. Dies geschieht mittels Lehrer-, Schüler- und Expertenmodell. ITS sollen hier aber nicht weiter betrachtet werden. Nähere Informationen zu diesem Thema findet man z.B. in der Dissertation [MAR04].

2.3.2 Aktuelle Konzepte

Eine Recherche zu Simulationen im E-Learning ergab, dass es aktuell keine etablierten E-Learning-Systeme (der Schwerpunkt liegt vor allem auf standardisierte Systeme, wovon zur Zeit nur SCORM in Frage kommt) gibt, die neben den üblichen multimedialen Inhalten auch Simulationen einsetzen. Zumindest keine, welche man sich normalerweise unter dem Begriff Simulation vorstellt. Bis jetzt sind Trainingssimulationen immer eigenständige Anwendungen, wie z.B. Flugsimulatoren, die nicht an ein Lernsystem mit heterogenen Lerninhalten angebunden sind.

Was man heutzutage finden kann, sind so genannte „Softwaresimulationen". Diese emulieren die Benutzung einer bestimmten Software und schulen damit die Bedienung dieser. Dazu gibt es zahlreiche Programme, die das Erstellen und Bearbeiten solcher Softwaresimulationen ermöglichen, und daraus sogar SCORM-konforme Lernmodule erzeugen, die in SCORM LMS genutzt werden können. Das dafür verwendete Medium ist üblicherweise Flash. Hinter einer Softwaresimulation verbirgt sich üblicherweise ein endlicher Automat, mit sensitiven Bereichen auf diversen Screenshots als Eingabe und einem nächsten Screenshot als Ausgabe und Folgezustand.

Weiterhin gibt es auch „Minisimulationen", die einfach genug sind, um z.B. als Flashprogramm realisiert zu werden. Allerdings dürften die Möglichkeiten einer solchen Simulation eingeschränkt sein. Außerdem stellt sich hier die Frage, inwieweit man noch von einer Simulation sprechen kann. Dagegen hat man den Vorteil, solche Programme ohne großen Aufwand über webbasierte Lernsysteme zu verbreiten. Der Nachteil solcher Stand-Alone-Anwendungen ist, dass sie nicht mit anderen Simulationssystemen kompatibel sind und wie eine Blackbox funk-

tionieren und daher kaum für die Bedürfnisse eines LMS angepasst werden können.

2.3.3 Simulationen als Lernmodule für LMS

LMS nach SCORM Standard bieten heutzutage nur die Möglichkeit, Standard Medien für Lerninhalte zu verwenden (Text, Bilder, Audio, Video) und diese nur geringfügig interaktiv zu gestalten. Im Gegensatz dazu werden hoch interaktive Simulationen nicht mit Lernumgebungen verknüpft, und die Zuteilung zu Lernenden und Auswertung von Simulationsdurchläufen von Hand durchgeführt. Das Ziel sollte sein, die Vorteile von LMS und Simulationen zu vereinen. Kurs- und Benutzerverwaltung, Adaptive Lernpfade in Kursen, Bewertung von Lernleistungen usw. sind Stärken von SCORM LMS, die Simulationssysteme im Allgemeinen nicht bieten. Für SCORM dagegen wären Simulationen als weiteres Medium in Lernkursen eine Bereicherung der Lernqualität, allerdings nur, wenn diese über standardisierte Schnittstellen angebunden werden – eigenständige, vom Standard abweichende Entwicklungen würden gegen das SCORM Prinzip arbeiten.

Diese Verschmelzung ist allerdings nicht trivial und bedarf weitreichender Untersuchungen. An dem Projekt arbeiten bereits einige Institutionen, zu lesen im nächsten Abschnitt 2.3.5 „Siumationsinterface – aktuelle Forschung". Ein erster Schritt in diese Richtung ist die Erstellung von Use-Cases. Diese repräsentieren typische Anwendungsbeispiele, woraus sich Anforderungen und Ziele der Arbeit ableiten lassen können.

Nach Brent Smiths (Mitglied der SCORM-Simulation Interface Standards Study Group) Statusbericht [SMITH] lassen sich zwei Sichten für eine Integration von SCORM und Simulationen unterscheiden: Lernkurs nutzt Simulation und Simulation nutzt Lernkurs (siehe Tabelle).

Lernkurs nutzt Simulation	**Simulation nutzt Lernkurs**
• Vorführung (Kurs steuert Simu zur Demonstration) • Ausprobieren • Test (Fähigkeitsbewertung) • LMS initialisiert Simulation und erhält Punkte zur Bewertung des Studenten	• Vorbetrachtung (Student lernt die für die Simu erforderlichen Kenntnisse) • Online Hinweise und Hilfe während der Simulationsausführung • Förderung (Nachbetrachtung der Leistung) • Speicherung (LMS nutzt Simulation)

Es zeigt sich hier eine gewisse Aufgabenverteilung: auf Seiten des LMS erfolgt die Vorbetrachtung und eine Art Debriefing für die Simulation sowie Auswertung und Speicherung von studentenspezifischen Daten im Rahmen der aktuellen Möglichkeiten von SCORM. Die Simulation dient als interaktive Testumgebung und Feedback der zuvor durch das LMS erläuterten Inhalte. Eine Trennung

in zwei Sichten, wie oben vorgeschlagen, macht wahrscheinlich wenig Sinn, weil im Idealfall mehrere Anwendungen beider Seiten in einen Lernkurs einfließen. Einige Punkte bedeuten bei genauerer Betrachtung auch dasselbe, nur aus anderer Perspektive, z.B. Vorführung / Vorbetrachtung oder LMS erhält Bewertungspunkte von der Simulation / Simulation nutzt LMS zur Speicherung von Bewertungen. Worin ein wirklicher Unterschied liegt wäre die Zuständigkeit der Steuerung. Das LMS bzw. der Kurs kann die Simulation für seine Zwecke kontrollieren, es kann aber auch die Simulation der Ausgangspunkt sein. Ein LMS als Initiator sehe ich für sinnvoller an, da hier die Verwaltung der ganzen Lernumgebung liegt (absolvierte Kurse, Bewertungen, vom Tutor vorgegebener Lernpfad, möglicherweise Portal zu weiteren Anwendungen usw.). Es folgt nun ein generischer Use Case, der den allgemeinen Ablauf eines Simulationskurses darstellt.

1. LMS liefert Kurs zum Studenten.
    1. Vorbereitendes Training / Test
    2. Erläutern von Kontext / Situation / Szenario <- Vorbetrachtung
    3. Definition der Lernziele
2. Kurs initialisiert Simulation.
    1. Definition der Anfangsbedingungen
    2. Simulation laden falls nötig
    3. Interface zum Nutzer bereitstellen
    4. Simulation auf Anfangsbedingungen setzen
3. Student interagiert mit Simulation.
    1. Option A: Student agiert in der Simulation <- Ausprobieren, Test
    2. Option B: Student beobachtet Simulation <- Vorführung
4. Kurs sendet Bewertung zum LMS. <- Speicherung
    1. Sofort
    2. Cached
5. LMS wählt und liefert nächsten Inhalt, wiederholend 1-5 nach Bedarf.

1. (1) Nachbetrachtung; (2) Weiterfördernder Kurs <- Förderung
2. (1) Neues Szenario; (2) Szenarioanpassungen
3. (1) Simulation stoppen; (2) Simulation läuft weiter

2.3.4 SCORM Simulation Interface – aktuelle Forschung

Derzeit arbeiten mehrere Gruppen an einem Konzept zur Integration von Simulationen in das SCORM. Das ist zum einen die ADL Simulations Working Group (zur Vereinfachung hier nur ADL Gruppe), einer Zusammenarbeit des Joint ADL Co-Laboratory3, ADL Co-Lab und DMSO (Defense Modeling and Simulation Office)4. Zum anderen die SCORM-Simulation Interface Standards Study Group5 des IEEE und SISO (zur Vereinfachung SISO Gruppe). Erstere ist Teil des ADL Network und somit auch mit starkem Fokus auf Wünsche des Department of Defense (DoD). Als Teil des DoD entwickelte das DMSO ein Simulationsinterface zur Synchronisation mehrerer heterogener Simulationen, genannt HLA (High Level Architecture). Das U.S. Militär und seine Partner setzen verstärkt auf Simulationen zu Trainingszwecken. Um aber die verschiedenen inkompatiblen Systeme effektiver zu nutzen, wurde HLA als eine Kommunikationsschnittstelle entwickelt, womit unterschiedliche Simulationen, die an den HLA Standard angepasst wurden, zusammenarbeiten können (z.B. Flug- und Panzersimulationen, die in eine Battlefield Simulation integriert werden). Es lassen sich auch Simulationen mit Realsystemen, also z.B. speziell für Simulationen modifizierte Schiffe, verbinden. Dadurch ergeben sich komplexere und realistischere Trainingsszenarien. Nähere Informationen zum Simulationssystem des DoD findet man in Marc Prenskys Buch: "Digital Game-Based Leraning – Chapter 10" [Prensky]. Die Bestrebungen der ADL Gruppe liegen in der Zusammenführung von SCORM und HLA. HLA Simulationen (vor allem militärische Simulationen) sollen in SCORM E-Learning Kurse integriert werden können, um die Trainingsanwendungen besser zu verwalten und Leistungen der Anwender besser auswerten zu können. Da sowohl ADL als auch DMSO vom DoD gegründet wurden und höchstwahrscheinlich direkt deren Interessen vertreten, dürfte eine eventuelle Anpassung des SCORM bzw. HLA Standards leichter durchgesetzt werden. Die zweite Gruppe arbeitet eher allgemein an einer Schnittstelle für Simulationen und Computerspielen zu SCORM, ohne dabei auf bestimmte Simulationen zu fokussieren. Inwieweit diese mit „ADL Gruppe" zusammenarbeitet, ist nicht bekannt. Geplant waren Face to Face Meetings zwischen SISO Gruppe und JADL. Aus der „ADL Gruppe" sind mehrere HLA/ADL Prototypen hervorgegangen, während die „SISO Gruppe" Topologien für Instruktionskomponenten in Simulation-LMS-Umgebungen erarbeitet hat. Auf beides wird im Abschnitt 3 näher eingegangen.

3 Konzeption

3.1 SCORM Einschränkungen und Herausforderungen

Obwohl bereits zwei große internationale Organisationen (siehe Abschnitt 2.3.5) seit mehreren Jahren an der Integration von Simulationen in SCORM arbeiten, sind die Entwicklungen bis jetzt noch nicht über einige Prototypen zu Testzwecken hinausgekommen. Dies liegt zum einen an der hier größeren beteiligten Interessengemeinschaft. Sowohl die SCORM-Community als auch die der Simulationen / HLA müssen Anforderungen und Interessen erarbeiten, validieren und auch durchsetzen bzw. mit anderen Gruppen auf einen gemeinsamen Nenner bringen. Dabei beteiligt sind Standardisierungsorganisationen, was die Anforderungen an das Ergebnis aufgrund der weitreichenden Folgen zusätzlich erhöht. Außerdem sind diese bemüht, existierende Standards gar nicht bzw. so geringfügig wie möglich zu verändern. Neben den konzeptionellen Schwierigkeiten ergeben sich auch technische Probleme bei der Integration, die darauf basieren, dass bei der Entwicklung von SCORM und Simulationen / HLA völlig verschiedene Anforderungen gestellt wurden. SCORM kann im Prinzip nichts anderes als webfähige Inhalte zu liefern, die durch eine (bei minimaler SCORM-Konformität nur wenig umfangreiche) API per Javascript und einem Applet über HTTP mit einem Server kommunizieren. Dieser auch Thinclient genannten Architektur stehen die Fatclients von Simulationen gegenüber. Es stellt sich die Frage, ob die auf Webpräsenzen ausgelegte SCORM-Architektur mit nicht-webfähigen Programmen sinnvoll verknüpft werden kann, ohne die Funktionalität der Simulationen oder die Mobilität von SCORM einzuschränken. Diese und weitere Fragen sollen bei der Erläuterung der drei grundsätzlichen Ansätze im Abschnitt 3.3 geklärt werden. Um aber die technischen Probleme besser zu verstehen, folgt ein kurzer Überblick über die Funktionsweise der LMS – Lernkurs Kommunikation.

Der Aufbau eines SCORM-Kurses wurde bereits in Abschnitt 2.1.4 behandelt. Hier geht es lediglich um die konkrete Umsetzung der Kommunikation zwischen diesem und dem LMS. Gewisse Details sind nötig, um die Grenzen von SCORM und dadurch die Schwierigkeiten bei der Integration von Simulationen aufzuzeigen.

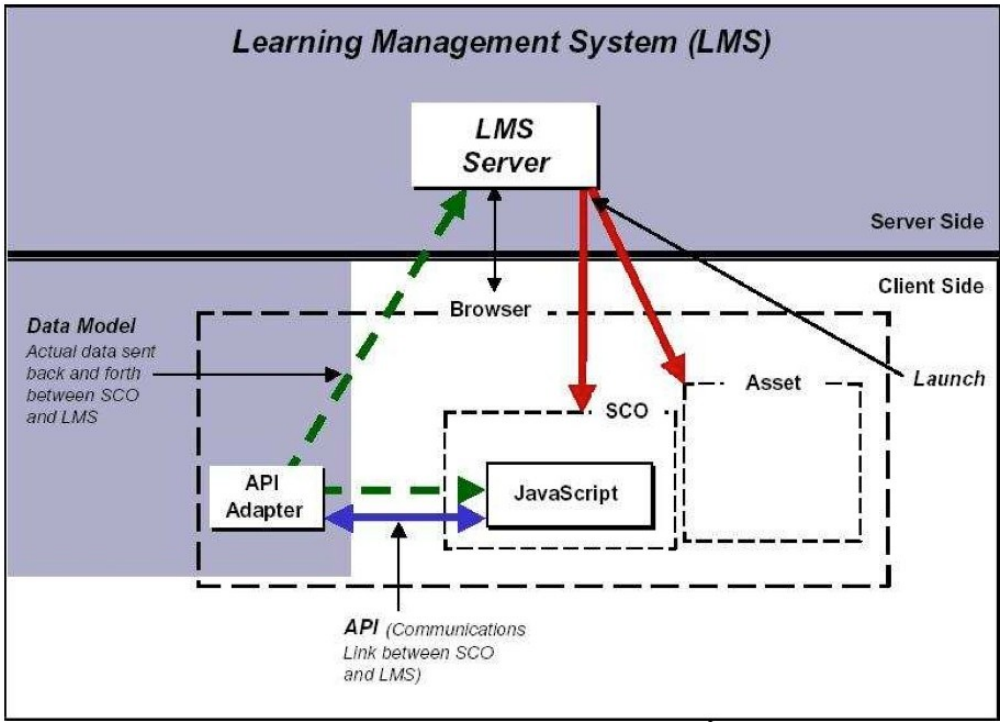

Abbildung 1: MLS-SCORM Kommunikation (ADL SCORM Version 1.2)

Eine mögliche Implementierung, wie sie z.B. von smartBLU verwendet wird, ist folgende. Der gesamte Kurs wird in einer HTML-Seite eingebettet und vom LMS geliefert. Darin integriert ist ein Applet und Javascript. Das Applet stellt die Verbindung zum LMS her, und der gesamte Datenaustausch zwischen Kurs und LMS erfolgt nur über diese eine Verbindung. Die Art und Weise der Kommunikation ist dabei dem Entwickler selbst überlassen. Es muss lediglich dafür gesorgt werden, dass API Requests vom Lernkurs korrekt verarbeitet werden. Die API, die durch die SCORM Runtime Environment bereitgestellt werden muss, ist als JavaSript realisiert. Dieses leitet aber lediglich die API Requests an das Applet weiter. SCORM spezifiziert also nur den Bereich API Adapter-SCO-Asset. Über den API Adapter hinaus, vor allem dem Bereich LMS, gibt es keine Spezifikationen. Einzig und allein das Verhalten auf die API Requests und die Bedeutung des Data Models ist vorgegeben. Sowohl das Applet als auch das JavaScript sind für alle Kurse eines LMS identisch. Lediglich die mitgelieferten SCOs und Assets mit dazugehörigem Content Aggregation unterscheiden sich. Diese sind es auch, die den Inhalt des Kurses ausmachen. Über die API können nur Daten ausgetauscht werden, die dem RTE Data Model entsprechen (siehe Abschnitt 2.1.4). Abweichungen würden eine nicht SCORM-konforme Erweiterung benötigen, die inkompatibel mit anderen SCORM LMS wäre. Es ist zwar offen, wie der API Adapter implementiert wird, und wie die Variable "API" einem SCO bereitgestellt wird, jedoch sind die grundsätzlichen Einschränken von SCORM gleich.

Aus der derzeitigen SCORM Architektur ergeben sich daher folgende Einschrän-

kungen bzw. Hindernisse:

1. Kommunikation nur unidirektional SCO
   In SCORM muss jede Kommunikation von einem SCO initialisiert werden. Das schreibt sowohl die Spezifikation vor, als auch die Verwendung von Applets. LMS können keine Verbindung zu einem SCO starten. Dieses Problem kann derzeit nur durch "busy waiting" oder regelmäßige Anfragen seitens des Applets zu einem Server umgangen werden.

2. Keine Inter-SCO-Kommunikation
   SCOs können und dürfen nicht direkt miteinander kommunizieren. Dieser Fakt lässt sich schon aus Punkt 1 ableiten. Es ist auch nicht möglich, mit anderen SCOs über das LMS Daten auszutauschen.

3. Keine Multiuserunterstützung
   Die SCORM Spezifikation sieht keine Multiuserunterstützung vor. Das heißt, dass gemeinsames bewältigen eines Lernkurses technisch nicht unterstützt wird. Jeder Teilnehmer muss seinen eigenen Kurs absolvieren, und wird auch nur aufgrund seiner eigenen Leistung bewertet, unabhängig von anderen Teilnehmern, die am gleichen Kurs teilnehmen. Für viele Trainingsszenarien im Simulationsbereich ist das völlig unzureichend. Externe Kommunikationsmittel wie z.B. Chats, wie sie meistens von LMS angeboten werden, bieten keine wirkliche Lösung für dieses Problem.

4. Nur ein User pro SCO
   Ein Kurs kann nur nutzergebunden gestartet werden. Alle Informationen, die der Kurs vom LMS erhält, oder die zum LMS übertragen werden (z.B. absolvierte Lernobjekte oder erreichte Punkte) sind immer auf den ausführenden Nutzer bezogen. Es können nicht mehrere Nutzer gleichzeitig mit einem SCO arbeiten.

5. Komplexität in einem SCO
   Ein SCO ist ein Webinhalt, und kann nicht beliebig mit ressourcenintensiven Funktionen ausgestattet werden und unterliegt gewissen Sicherheitsbestimmungen auf Datei- und Prozessebene.

6. Fehlende Metadaten für Simulation
   Ziel von SCORM ist unter anderem die leichte Verbreitung und Wiederverwendung von Lerninhalten. Wenn die Nutzung von Simulationen über SCORM ähnlich erfolgreich sein soll, muss dieses auch für Simulationen möglich sein. Dazu bedarf es aber neuer Metadatenelemente, da die bisherigen Simulationen sicherlich nicht adäquat genug beschreiben können.

7. CMI Datenmodell nicht ausreichend für Simulationsscores
   Eine wichtige Funktion für Simulationen, die vom DMSO gefordert wird und daher ein SCORM LMS auch unterstützen werden muss, ist eine nutzerunterstützende Instructional Component (siehe Abschnitt 3.5). Teil dieser Komponente wird unter anderem das „Debriefing" sein. Und da allgemein Trainingsszenarien Teil der E-Learning-Simulation-Welt sein sollten, dürfte diese Funktion auch für andere Anwendungen sinnvoll sein. Mit dem bisherigen CMI Datenmodell lassen sich aber die Aktivitäten in einer Simulation nicht ausreichend genug auswerten. Ein weiteres Problem wird sein, dass jede Simulation andere Ereignisse und Daten liefert.

8. Abspeicherung des Simulationsfortschritts
   Für das Unterbrechen und spätere Fortsetzen von Kursen existiert ein Bookmarking-Mechanismus, der aber nur IDs von Seiten innerhalb des Kurses speichert. Für Simulationen müsste dieser Mechanismus erweitert werden, oder ein weiterer hinzugefügt werden. Es werden komplexere Daten für die Erfassung eines Zustandes in der Simulation benötigt.

9. Problem mit Fatclient Simulationen
   Optimal wäre es, alle Simulationen über Webbrowser durch das LMS nutzen zu können. Ob dies aber mit allen Fatclients von Simulationen möglich sein wird, ist fraglich. Thinclients werden sich dagegen erwartungsgemäß leichter integrieren lassen. Man muss allerdings bedenken, ob es sinnvoll ist, Simulationen über mehrere Konzepte gleichzeitig zu integrieren. Die nächste Frage bei Fatclients wäre, wie sie dem Nutzer zugänglich gemacht werden, wenn nicht über den Browser. Und wie die lokal laufende Simulation eine Verbindung zum LMS oder Lernkurs im Browser herstellt.

3.2 Mögliche Anforderungen

Welche Anforderungen können oder müssen an ein SCORM-Simulations-Interface gestellt werden? Einige davon lassen sich direkt aus den Einschränkungen (Abschnitt 3.1) ableiten. Die folgende Liste nennt zunächst alle Anforderungen ungeachtet der Erfüllbarkeit.

1. Keine oder möglichst geringe Änderungen am SCORM Standard
   Jede Veränderung am SCORM Standard kann zu Inkompatibilitäten mit bestehenden SCORM Systemen führen und sollte daher wenn möglich vermieden werden.

2. Keine oder möglichst geringe Änderungen an der Simulation
   Gleiches gilt für Simulationen. Die Anforderungen an eine Simulation, um mit einem SCORM Interface kompatibel zu sein, sollten möglichst

gering gehalten werden. Je höher die Anforderungen, desto größer der Anpassungsaufwand, und desto kleiner wird das Spektrum an nutzbaren Simulationen.

3. Simulationsneutrales Konzept
   Ein wichtiger Punkt sollte sein, dass bei der Konzeption nicht von Simulationsstandards ausgegangen wird. Bei der Forschung der „ADL Simulations Working Group" scheint es primär um HLA als Simulationsstandard zu gehen. Eine Beschränkung auf HLA würde die Sache zwar vereinfachen, es müsste dann aber viel Aufwand betrieben werden, um andere Simulationen HLA kompatibel zu machen.

4. Thinclient Architektur für Simulationen ermöglichen
   Idealerweise sollten Simulationen wie alle anderen SCORM Lerninhalte auch über den Webbrowser an jedem Ort abrufbar sein. Lokale Installationen sind eher ein Hindernis.

5. Learning Objects für Simulationen
   Es müssen für Simulationen spezifische Lernobjekte (Lernziele in einem Kurs) angegeben werden können. Da diese sich von bisherigen Lernobjekten in SCORM unterscheiden werden, müssen diese aufeinander abgebildet werden oder die Kapazitäten der Lernobjekte erweitert werden.

6. Zwei-Wege-Kommunikation (Nutzer – Simulation) für interaktive Instruktionskomponente
   Eine interaktive Instruktionskomponente (siehe Abschnitt 3.5), die Informationen in beide Richtungen, vom Nutzer zur Simulation und anders herum, austauscht, benötigt Kommunikationswege in beide Richtungen. Diese können über das LMS oder das SCO gehen.

7. Tracking und Erfolgskontrolle
   Aktivitäten des Nutzers in der Simulation sollen aufgezeichnet und später ausgewertet werden können, entweder durch Analyse der Lernobjekte und Scores, oder Rekonstruktion des Simulationsablaufs.
   Die Leistungsbewertung eines Nutzers muss anhand seiner Aktivitäten in der Simulation möglich sein.
   Die Simulation muss relevante Daten für eine Erfolgskontrolle liefern. Möglicherweise ist ein neues Score-Modell notwendig.

8. Sharing von Simulationslernkursen
   Kurse, die auf Simulationen zugreifen, sollen genauso leicht zwischen LMS ausgetauscht werden können, wie normale SCORM Kurse. Es werden daher Metadaten zur Beschreibung eines Simulationskurses benötigt, und Mechanismen, Simulationen einheitlich an ein LMS anzumelden.

9. Kollaborative Learning
   Es sollen Lerninhalte gemeinsam von Lernenden bearbeitet werden können. Aktivitäten eines Teilnehmers von einem kooperativen Kurs sollen für andere sichtbar sein und auch Auswirkungen auf deren Aktivitäten haben, je nach Szenario des Lernkurses. Andere Aspekte sind z.B. kursinterne Kommunikationswerkzeuge.

3.3 Drei Integrationskonzepte SCORM-HLA

Von der **ADL Simulations Working Group** gibt es drei Konzepte, bezeichnet als Class 1, Class 2 und Class 3. Diese werden in diesem Abschnitt erläutert und bewertet. Generell steigt die Komplexität der Konzepte von Class 1 zu Class 3, sowie die Veränderungen am Standard und die daraus resultierenden nötigen kritischen Betrachtungen.

3.3.1 Class 1 Konzept

Class 1 setzt bei der Integration am SCO an. Wie in der folgenden Abbildung (Abbildung 2) zu sehen ist, werden das SCORM Interface sowie die Simulation (hier HLA) werden klar durch das SCO getrennt, welches als Schnittstelle dient zwischen beiden dient. Das bedeutet zwar zum einen, dass keine Veränderungen am SCORM Standard notwendig sind, aber zum anderen auch, dass eventuell sehr viel Komplexität in einem SCO implementiert werden muss. Die Ausführungen des DMSO zum Class 1 Prototypen können so verstanden werden, dass Veränderungen am SCORM Standard nicht nur nicht notwendig, sondern auch keinesfalls durchgeführt werden sollen. So ausgelegt bedeutet das umso mehr Funktionalität im SCO, da davon ausgegangen werden muss, dass die „SCORM Conduits“ und „Simulation Conduits“ völlig inkompatibel zueinander sind und das SCO als Puffer und Übersetzer arbeitet.

Die Funktionsweise von Class 1 sieht dann folgendermaßen aus: Zwischen SCO und LMS wird nur „SCORM gesprochen“ (die SCORM Conduits). Das SCO wird vom LMS in den Browser geladen, und kommuniziert anschließend über die SCORM API und das SCORM Data Model. Außerdem nimmt das SCO Verbindung mit der Simulation (hier HLA) auf über die simulationsspezifische API. Das kann auch indirekt über weitere (LMS-externe) Komponenten geschehen.

Aus dieser Konstellation und der Tatsache, dass SCORM nicht modifiziert werden soll, lassen sich mit den vorhandenen SCORM-Möglichkeiten folgende Funktionen und Eigenschaften ableiten:

1. Keine SCORM Modifikationen

2. Wahrscheinlich sind Modifikationen an der Simulation notwendig (Bewertung und Erfolgskontrolle ermöglichen, Bookmarking des Zustandes

ermöglichen, Ereignisse für Tracking ermöglichen).

3. Lernziele für eine Simulation müssen auf SCORM Learning Objects abgebildet werden (cmi.objectives).

4. Tracking und Erfolgskontrolle ist über cmi.interactions möglich, benötigt aber diskrete Ereignisse von der Simulation. Es ist keine kontinuierliche Aufzeichnung des Simulationsverlaufs möglich.

5. Eine Steuerung der Simulation durch das LMS ist nicht möglich, wohl aber durch ein SCO im Lernkurs.

6. Zur Speicherung des Zustands der Simulation im LMS steht nur cmi.core.lesson_location (255 Zeichen) zur Verfügung. Dies ist nicht ausreichend für Persitenzdaten.

7. Multiuser nicht möglich (generell unmöglich ohne SCORM Modifikation)

8. Möglicherweise ressourcenintensives SCO

9. Das Sharing des Kurses bleibt problemlos, solange der Verbindungsaufbau zwischen Kurs und Simulation LMS-unabhängig ist.

Folgendes Bild zeigt das Class 1 Konzept am Beispiel von HLA

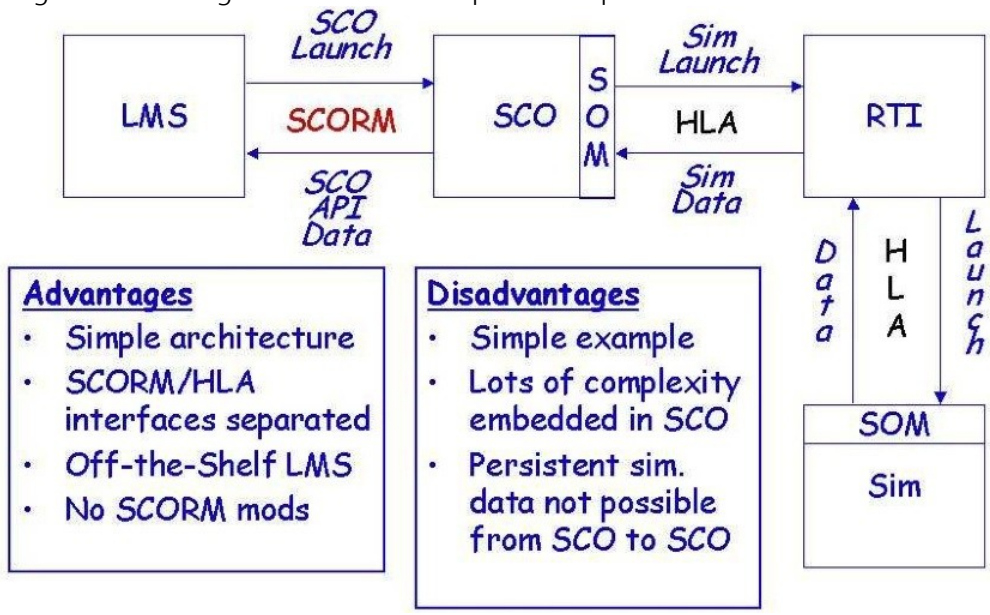

Abbildung 2: Class 1 Prototype

Zum Class 1 Konzept gibt es bereits einen (oder mehrere) Prototypen. Wie in der nächsten Abbildung zu sehen ist, steht zwischen der Simulations-Runtime (RTI) und dem LMS der SCORM Kurs (alle Elemente im Browser), dargestellt durch das aktuelle Collector SCO und einem Launcher Asset. Das Launcher Asset wird nur einmal benutzt, um eine externe „Listener Application" – verbunden mit der Simulations-Runtime – die Simulation (optional mit Parametern wie Learning Objects) zu starten. Die Kommunikation zwischen Kurs und Simulation erfolgt dann über ein Collector Applet, welches im Kurs integriert ist. Dieses versorgt die Simulation mit weiteren Daten, die im LMS abgespeichert sind (z.B. frühere Ergebnisse des Studenten) und nimmt Ereignisse und Ergebnisse der Simulation entgegen. Diese können dann in SCORM-kompatible Daten (z.B. Interactions, Score) umgewandelt und wieder im LMS gespeichert werden. Der API Wrapper ist hier vermutlich die unveränderte SCORM API (die ja nicht modifiziert werden soll).

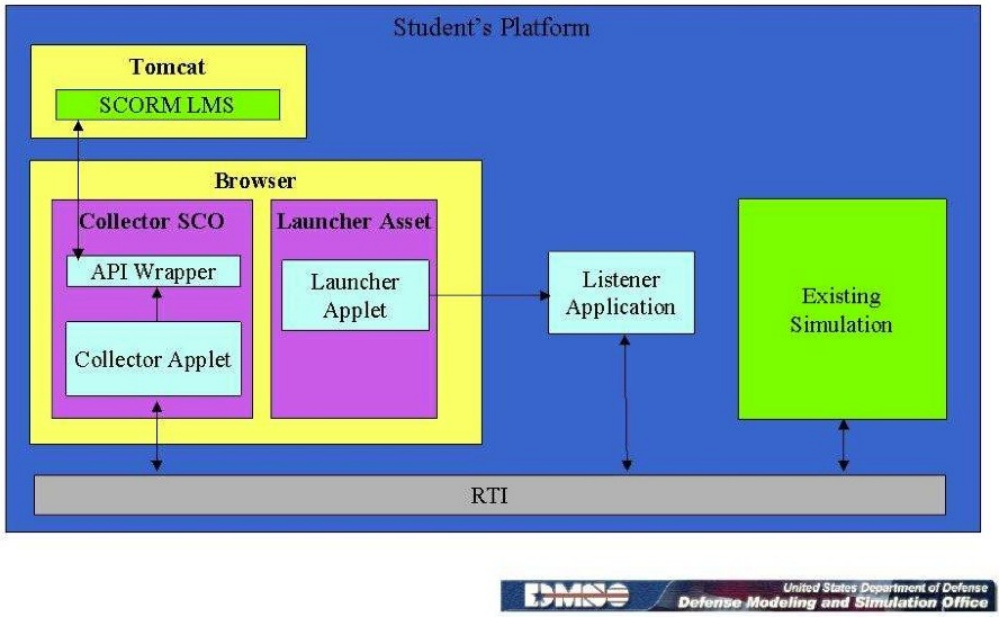

Abbildung 3: DMSO Initial HLA/ADL Interface

Anwendung und Empfehlung:

Der große Vorteil bei Class 1 liegt in der leichten Portierung. Jedes SCORM LMS wäre kompatibel zu einem Simulationskurs, solange die Simulation für den Kurs erreichbar ist. Der Nachteil liegt allerdings darin, dass im Prinzip jeder neue Kurs, der mit einer Simulation arbeiten soll, von Grund auf neu gestaltet werden muss und die gesamte Funktionalität für die Simulation-Kurs-Kommunikation implementieren muss. Außerdem sieht diese Implementierung für jede andere Simulation auch anders aus. Die Erstellung eines solchen Kurses dürfte also mit sehr viel Aufwand verbunden sein. Aufgrund der Einschränkungen des SCORM Datenmodells und der API eignet sich dieses Konzept auch nicht für alle Simulationen. Je

interaktiver der Ablauf und komplexer die erzeugten Daten dabei sind, desto schwieriger und unübersichtlicher wird eine Abbildung auf das SCORM Modell. Typ 1 Simulationen (siehe Abschnitt 2.2.5) lassen sich so sicher ohne Probleme nutzen, da diese ganz auf Interaktivität verzichten, Typ 2 mit der Einschränkung, dass das Tracking und Pausieren / Stoppen auf einfachen Daten basieren muss. Typ 3 lässt sich überhaupt nicht empfehlen, da sich ein Multiuserkurs nicht über ein normales SCORM LMS realisieren lässt. Ein Student kann sich nicht mit einem laufenden Kurs oder SCO eines anderen verbinden, und Inter-SCO-Kommunikation ist ebenfalls nicht möglich. Über den Umweg durch die Simulation ließe sich das Problem beheben, wenn diese dann die Synchronisation der Teilnehmer übernimmt und studentenbezogene Daten an die jeweiligen SCOs übermittelt. Dann stellt sich aber die Frage, ob man die Kontrolle eher bei der Simulation oder dem LMS überlassen möchte. In Hinblick auf zukünftige weitere Erweiterungen von SCORM empfiehlt sich klar das LMS.

3.3.2 Class 2 Konzept

Das zweite Konzept basiert auf dem Ersten, und gleicht diesem auch größtenteils. Wieder ist das SCO bzw. der Lernkurs der Point of Integration, wie in der folgenden Abbildung (Abbildung 4) zu sehen. Allerdings sollen einige Nachteile aus Class 1 behoben werden, indem eine Erweiterung des SCORM Datenmodells zur Speicherung von Persistenzdaten der Simulation erlaubt wird, oder alternativ das SCO über LMS-externe Dienste die Möglichkeit hat, Daten in selbst definierten Formaten abzuspeichern. Indirekt ist so auch die Inter-SCO-Kommunikation möglich, und Simulationsdaten könnten von einem SCO zum anderen übertragen werden, z.B. zum Pausieren und Fortsetzen von Simulationen.

Die folgende Abbildung zeigt das Class 2 Konzept. SCORM und Simulation sind wieder durch das SCO voneinander getrennt. Dadurch ergibt sich wie in Class 1 viel Komplexität im SCO, das wieder als Schnittstelle dient. Optional können SCOs auf externe Datenbanken zugreifen, um SCORM Einschränkungen zu umgehen.

Ein früher Class 2 Prototyp vom DMSO nutzt eine spezielle Webserviceanwendung, um Daten von der HLA RTI zu sammeln und sie dem SCO zugänglich zu machen. Webservices können leicht an die eigenen Bedürfnisse angepasst werden, und das verwendete HTTP vermeidet Netzwerkprobleme, wie z.B. Firewallsperren. Diese konnten bei diesen Tests aber sowieso nicht auftreten, da alle Komponenten (LMS, HLA-RTI, Web Service Plattform) auf demselben Rechner liefen, zu Testzwecken. Aus der Veröffentlichung geht hervor, dass Tests zu remote zugreifbaren Simulationen gemacht werden sollen / wurden. Wie weit diese fortgeschritten sind, oder welche Ergebnisse diese ergaben, ist nicht bekannt. Das Problem der notwendigen clientseitigen Installation der Simulation besteht im Übrigen auch bei allen anderen Konzepten. Diese machen zu diesem Thema sind bisher auch keine Informationen zu finden. Die folgende Abbildung zeigt

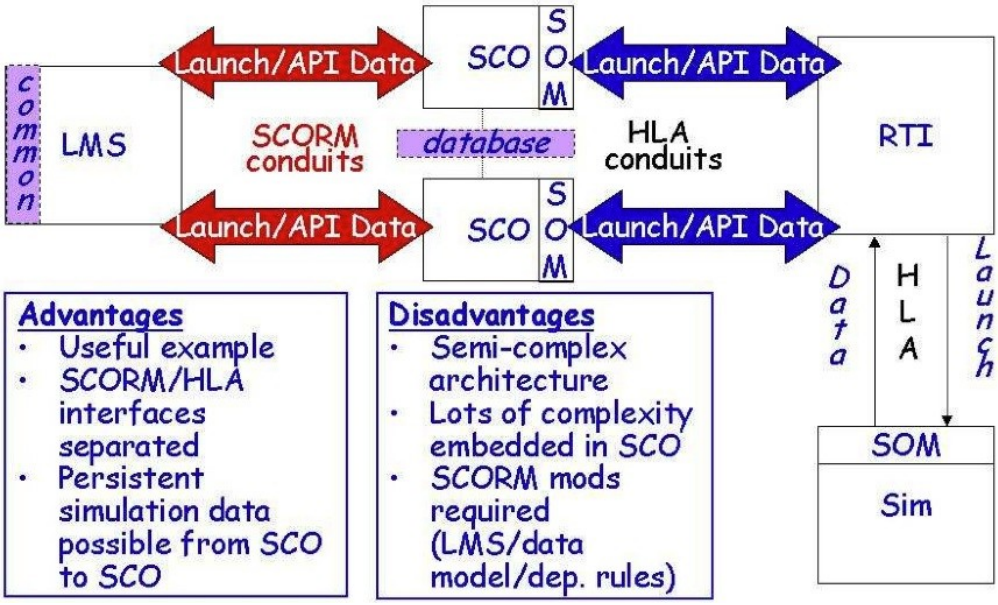

Abbildung 4: Class 2 Prototype

den DMSO Class 2 Prototypen.

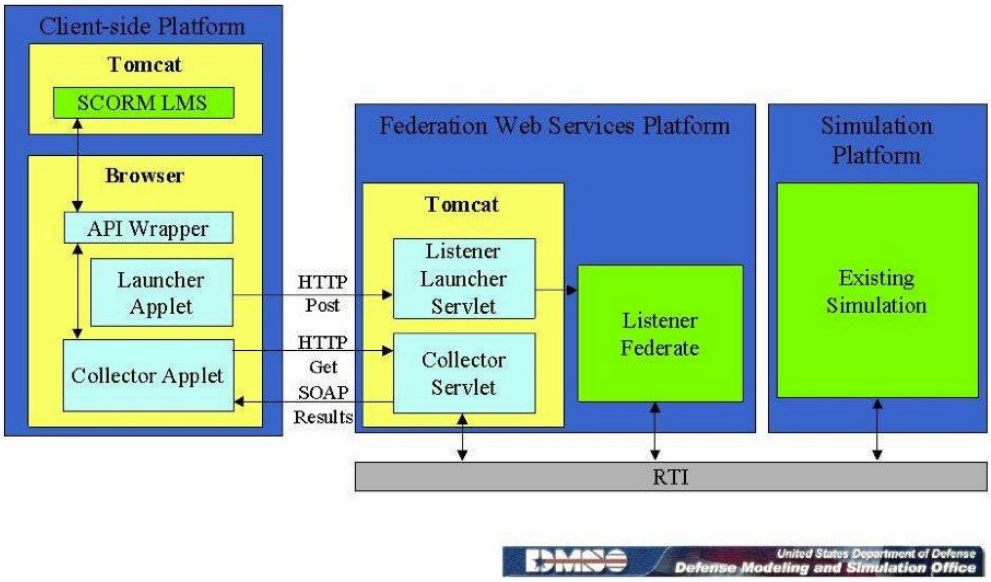

Abbildung 5: DMSO SOAP-based HLA/ADL Architecture [ADL/HLA1]

Anwendung und Empfehlung

Class 2 bietet mehr Möglichkeiten für komplexere Simulationsdaten als Class 1, verlangt aber auch entweder eine Erweiterung des SCORM Datenmodells, oder die Entwicklung von externen Diensten für SCORM-untaugliche Simulationsda-

ten. Bei beiden Varianten sollte auf Neutralität gegenüber Simulationen geachtet werden, um diese kompatibel für verschiedene Simulationssysteme zu gestalten. Falls man auf einen simulationsuniversellen externen Dienst verzichtet, kann man auch leichter und schneller simulationsspezifische Varianten erstellen. Hier ergibt sich dann aber das gleiche Problem wie bei der direkten Verknüpfung von SCO und Simulation, dass diese Komponente für jedes Simulationssystem neu entwickelt werden muss. Ansonsten bleiben beim Class 2 Konzept ähnliche Mängel, wie bei Class 1. Komplexe und teuer zu entwickelnde SCOs, keine direkte Kontrolle der Simulation durch das LMS.

Für Simulationen vom Typ 1 (siehe Abschnitt 2.2.5), die wenig Anspruch an eine Integration stellen, wäre der Aufwand einer SCORM Modifikation oder die Entwicklung einer Serviceplattform zu groß. Klasse 2 Typen würden hier deutlicher profitieren. Aus den DMSO Unterlagen ist nicht ersichtlich, ob die Serviceplattform die Simulationsdaten in irgendeiner Form aufbereitet. Das würde jedenfalls zu einer einfacheren und ressourcenschonenderen Interaktion zwischen SCO und Simulation führen. Für Klasse 3 Simulationen gilt das gleiche wie bei Class 1 Prototypen, dass Multiuser zwar möglich wäre, aber ohne Kontrolle des LMS. Durch die Serviceplattform gebe es aber immerhin schon eine zentrale Stelle, die für Kommunikation und Synchronisation der SCO zuständig wäre.

Für eine Standardisierung wäre Konzept 2 eine erste Basis. Der Prozess muss aber über die bloße Erweiterung des Datenmodells hinausgehen. Die Kommunikation zwischen SCO und Serivce Platform, die Funktionen der Service Platform sowie API und Datenmodelle sollten mit inbegriffen sein. Das dieser Prozess ein sehr langwieriger sein wird, sollte klar sein, allerdings laufen Tests mit diesem System schon seit mehreren Jahren. Die Ergebnisse sind leider unbekannt.

3.3.3 Class 3 Konzept

Das Class 3 Konzept ist Grundlegend anders. Die Verbindung von SCORM und Simulation geht nicht über das SCO, sondern das LMS selbst, bzw. einem speziellen Simulationsinterface, das sowohl SCORM sprechen kann, als auch HLA (oder jede beliebige andere Simulation).

Da die SCOs nicht direkt mit der Simulation in Kontakt kommen, aber Daten für den Kurs bentigt werden, muss die SCORM API und das Datenmodell erweitert werden. Diese sollte umfassend genug sein, um alle Use Cases abdecken zu können. Benötigt wird z.B. Simulation starten / pausieren / fortsetzen / beenden, Scores oder Ereignisse der Simulation abzufragen und zu speichern.

Über diese erweiterte API können dann Anfragen von SCOs an bestimmte Simulationen gestellt werden. Da das LMS diese zunächst verarbeitet, bevor sie an das Simulationsinterface weitergeleitet werden, hat es in diesem Konzept auch

Kontrollmöglichkeiten. Die großen Unbekannten scheinen im Moment noch das SIS/SOM (Simulationsinterface) und S3Data zu sein. Fest steht, dass hier wie auch in den anderen beiden Konzepten zuvor Simulationsdaten und Ereignisse für das LMS gesammelt werden. Ob diese nun aber schon SCORM-konform (auf die neue API und Datenmodell) im SIS/SOM umgewandelt werden, oder erst im LMS, ist nicht ersichtlich. Zum SIS gibt es, wie im Abschnitt 2.3.5 bereits erwähnt, Forschungen seitens AICC. Danach soll das SIS eine Instructional Component (oder auch Assessment Modul) enthalten. Diese Komponente steuert die Simulation nach bestimmten Anweisungen des Kurses, und kann den Studenten beim Simulationsverlauf unterstützend begleiten, durch z.B. eingeblendete Hilfestellungen, Missionsziele usw..

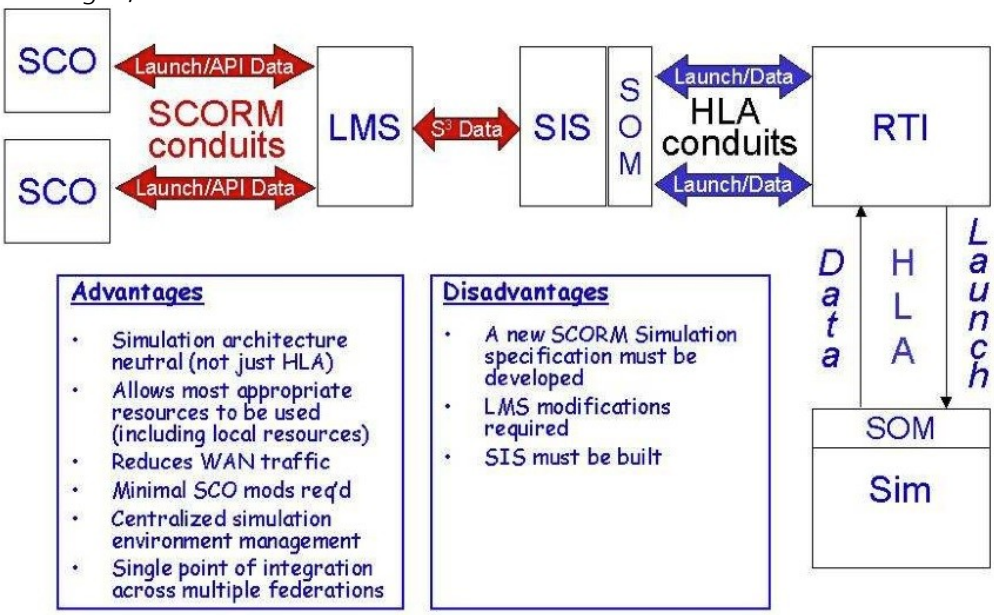

Abbildung 6: Class 3 Prototype

Anwendung und Empfehlung

Dieses Design hat das größte Potential, Simulationen vollständig in die SCORM-Welt zu integrieren. Da SCORM auf jeden Fall erweitert werden muss, kann man auch gleich alle benötigten Komponenten integrieren. Das wären ein neues Datenmodell, API, und die Möglichkeit des LMS, Anweisungen an ein SCO zu geben. Und früher oder später wird der Multiuseraspekt auch Thema für einen neuen SCORM Standard werden. Das Problem sollte dann besser jetzt am konkreten Beispiel Simulation angegangen werden. Wenn daraus ein Konzept entsteht, profitieren auch normale SCORM Kurse davon.

Während bei Class 1 und 2 der größte Aufwand bei den Entwicklern von SCOs lag, kann hier auf eine definierte API und Datenmodell (zugeschnitten auf Simu-

lationen) zugegriffen werden. Das reduziert die Kosten für Lernkurse erheblich, da die gesamte Funktionalität zur Kommunikation mit der Simulation und zur Umwandlung auf SCORM Daten wegfällt. Außerdem wird der Datenverkehr zwischen User und LMS reduziert.

Die Liste der Vorteile scheint lang, jedoch tut man sich beim DMSO mit dem Class 3 Konzept schwer. Das zeigt sich schon daran, dass es zum Zeitpunkt der Veröffentlichung [ADL/HLA1] noch keinen Prototypen gab, und dass es bis heute keine Veröffentlichungen über Ergebnisse an Class 3 Forschungen gibt. Die Bestimmung der Anforderungen an einen neuen SCORM Standard ist sehr aufwendig. Dann gehen Vorschläge durch ein Standardisierungsverfahren. Dieser Prozess kann Jahre in Anspruch nehmen. Und es ist nicht sicher, ob sich der immense Aufwand lohnt, und ein neuer Standard von der Community akzeptiert wird.

Trotzdem ist Class 3 immer noch der geeignetste Kandidat für Klasse 3 Simulationen. Eine zentrale Integrationsstelle für die Simulation ist besser für kollaborative Kurse, als selbsterstellte externe Dienste, welche keinen Kontakt zum LMS haben, und höchstwahrscheinlich zu anderen SCORM LMS inkompatibel sind. Klasse 1 und 2 Simulationen eignen sich natürlich auch für Class 3 Konzept, aber wie schon im Class 2 Konzept gesagt, würde der Entwicklungsaufwand in keinem Verhältnis zum Nutzen stehen.

3.4 Instructional Component

Die Forschungen seitens AICC haben einen anderen Ansatz. Statt konkreter Konzeptvorschläge wird eine „Instructional Conponent“ als Schnittstelle zwischen LMS, Simulation und Student angenommen. Diese Komponente ist „designed to teach“, und nutzt die Simulation zur Bereitstellung von Szenarien, stellt Anfangsbedingungen ein, beobachtet die Aktivitäten des Studenten, kann Fehler melden usw. Es geht nun darum, in welchen möglichen Topologien die Instructional Component, die Simulationsengine, das LMS und das Interface (einfach nur die Eingabestelle für den Studenten) angeordnet sein können. Als Ort für diese vier Komponenten gibt es lokale Rechner und Netzwerke. Die sieben Topologien, die bestimmt wurden, dürften im Allgemeinen die Bedürfnisse der Anwender widerspiegeln. Ziel der Auflistung der Topologien soll es sein, auf dieser Basis Anforderungen an ein SCORM Simulationsinterface genauer zu bestimmen.

Es sollen hier nicht alle Topologien erläutert werden. Dazu sei auf die AICC Veröffentlichung “Simulation-Based Instructional Systems Components and Topologies” [AICC] verwiesen. Die folgende Abbildung zeigt Topologie 1, anhand derer das Prinzip erklärt werden kann.

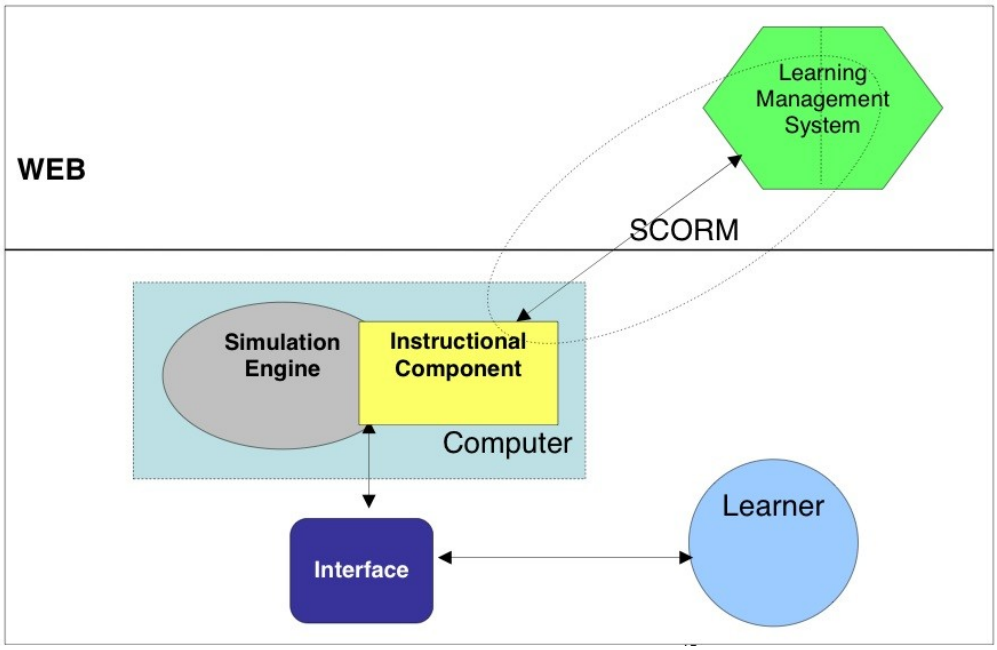

Abbildung 7: AICC Topologie

Das LMS wird über das Web erreicht, alle anderen Komponenten liegen auf dem lokalen Rechner. Zwischen LMS und Simulation liegt in jeder Topologie die Instructional Component, welche hier in der Simulation Engine bereits integriert ist. In anderen Varianten ist es eine eigenständige Komponente, die auch im Web oder auf anderen lokalen Rechnern liegen kann. Der Lernende arbeitet über das Interface nicht direkt mit der Simulation, sondern mit der Instructional Component. Auch hier gibt es andere Möglichkeiten, z.B. dass das Interface direkt an die Simulation gekoppelt ist, oder dass es ein Interface zur Instructional Component, und eins zur Simulation gibt.

Laut der AICC muss jede Topologie mit einem möglichen SCORM Simulationsinterface bedient werden. Dazu ist Interoperabilität in der SCORM- und Simulationswelt notwendig. Für SCORM ist der bedeutenste Aspekt, dass jedes SCO mit jedem LMS arbeiten muss. Dies wurde durch standardisierte Verfahren erreicht:

1. zum Starten eines SCO
2. eine IEEE API zur Kommunikation zwischen LMS und SCO
3. ein IEEE Datenmodell als Datensatz
4. CMI Format für das Datenmodell

Zur Interoperabilität seitens der Simulationen konzentriert sich AICC auf folgende zwei Punkte:

1. jede Simulation mit jedem grafischen Userinterface arbeiten zu lassen

2. jede Simulationseninge mit jeder Instructional Component arbeiten zu lassen

Die Beschränkung auf diese beiden Aspekte sollte genügen, da die anderen benötigten Kommunikationswege (Instructional Component – LMS) bereits durch SCORM (siehe Abbildung 6) standardisiert sind, und eine Erweiterung der SCORM API / des Datenmodells keine technische Umstellung verlangt. Um nun jede Simulationsengine mit jeder Instructional Component arbeiten zu lassen, müssen ähnliche Bedingungen wie bei LMS – SCO geschaffen werden:

1. Standard Verfahren für eine Instructional Component, eine Simulation zu starten
2. Standard Verfahren für eine Instructional Component, mit einer laufenden Simulation zu verbinden
3. Standardisiertes Kommunikationsverfahren zwischen Simulation und Instructional Component (API)
4. Standarddatensatz für den Informationsaustausch (Datenmodell)
5. Standardformat für die Daten (XML wird von AICC bevorzugt)

Leider sind in der AICC Veröffentlichung zur Instructional Component keine konkreten Schlüsse zu möglichen Konzepten ersichtlich. Die einzigen Feststellungen, die getroffen wurden, sind, dass es anscheinend jeder Simulationslernkurs eine Instructional Component benötigt, um dem Lernenden lernfördernde Maßnahmen mitzuteilen. Weiterhin, dass die vier Hauptkomponenten in einer LMS-Simulation-Umgebung, nämlich LMS, Simulation Engine, Instructional Component und Interface praktisch beliebig lokal oder im Web verteilt sein können. Aus diesem Grund sollte also immer von einer Kommunikation über das Web ausgegangen werden. Und als letzten Punkt wird eine mögliche Simulations-API in Betracht gezogen, die ähnlich der SCORM API funktionieren soll.

4 Umsetzung

In diesem Kapitel wird zunächst die verwendete Simulation kurz beschrieben, anschließend die Wahl des Konzepts begründet, und schließlich wird auf die Umsetzung des Konzepts durch SCORM Erweiterungen und SCORM externe Dienste eingegangen.

4.1 Simulation: Game of Life

Game of Life ist eine selbst entwickelte Simulation und stellt stark vereinfacht das Wachstumsverhalten von Zellen dar. Modelliert wurde dieses mit einem zellulären Automaten. Eine Zelle im Automaten stellt hier eine biologische Zelle dar. Deren Zustände können „lebendig“ und „tot“ sein. Der Zustandsübergang ist abhängig von zwei Regeln. Erstens, wann eine lebende Zelle stirbt, zweitens, wann eine tote Zelle belebt wird. Die Regeln sind in den Zellen verankert und besagen, wie viele lebendige und tote Zellen sich in der Nachbarschaft befinden dürfen / müssen, damit die Übergänge eintreten. Die Ausgabe einer Zelle ist deren Zustand, weis für tot, schwarz für lebendig. Der Simulationsablauf erfolgt in Zyklen. In jedem wird nacheinander der Zustandsübergang jeder Zelle überprüft und zwischengespeichert, bevor der globale Zustandsübergang des Zyklus erfolgt.

Um verschiedene Effekte des Game of Life zu zeigen, kann die Simulation mit unterschiedlichen Szenarien und Regelsätzen initialisiert werden. Das Szenario bestimmt die Anfangsverteilung der Zellen. Zum Beispiel zufällig, oder in Blöcken gleicher Zustände. Die Veränderung der beiden Regeln führt zu einem unterschiedlichen Wachstum. So können sich bei ausreichend vielen Zyklen stabile Muster ergeben, oder die gesamte Zellpopulation verschwinden.

Die folgende Abbildung zeigt Game of Life im Startzustand mit einer zufälligen Verteilung der Zellen mit 30% lebendigen Zellen. Die zweite Abbildung die gleiche Szene nach Ausbildung einer festen Anordnung.

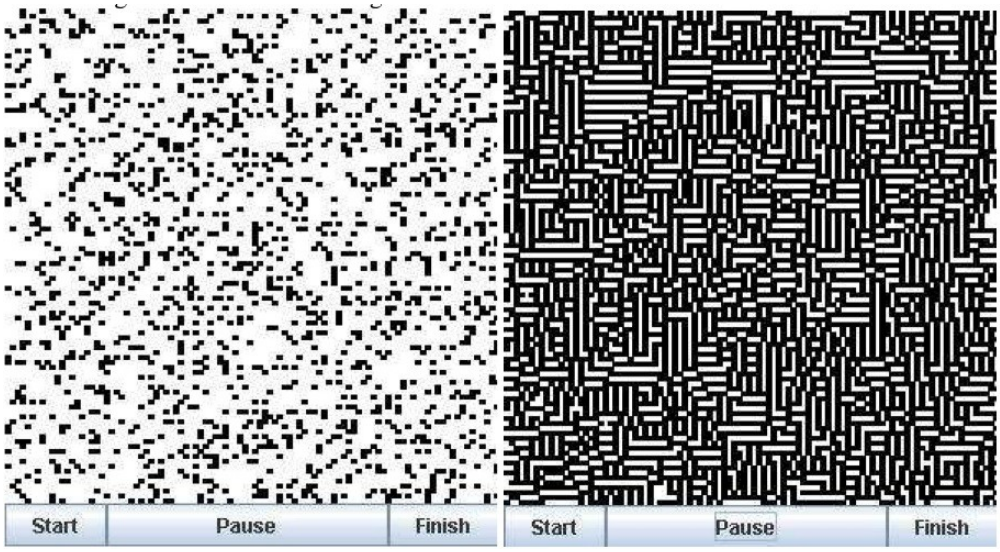

Abbildung 8: Game of Life: Startzustand und Endzustand

Die verfügbaren Funktionen:

1. Start – startet die Simulation oder setzt sie nach einer Pause fort
2. Pause – pausiert den Simulationsverlauf
3. Finish – die Simulation wird gestoppt und kann nicht wieder fortgesetzt werden

In Simulationslauf werden gewisse Daten generiert, die später vom LMS zu erfassen sind. Diese sind natürlich beliebig erweiterbar. Beispielhaft wurden folgende Daten gewählt.

Daten der Simulation:

1. User
2. Datum
3. Szenario
4. Endpopulation der Zellen (prozentualer Anteil der lebendigen Zellen)
5. Regel 1
6. Regel 2

Dabei werden lediglich Szenario, Endpopulation, Regelsatz 1 und 2 von der Simulation geliefert. Der User wird durch den Lernkurs über die SCORM API bestimmt, und das Datum automatisch gesetzt.

4.2 Wahl des Konzepts

Zur prototypischen Integration einer Simulation in SCORM wurde das Class 2 Konzept gewählt. Dieses bietet wahrscheinlich die beste Balance zwischen Implementierungsaufwand und SCORM-Kompatibilität auf der einen Seite und Erweiterbarkeit für Simulationsdaten auf der anderen. Die genauen Vor- und Nachteile jedes der drei Konzepte können im jeweiligen Abschnitt im Kapitel 3 nachgelesen werden. Zusammenfassend kann man sagen, dass Class 1 zu unflexibel für Erweiterungen ist aufgrund der Tatsache, dass SCORM dort nicht verändert werden soll. Zum speichern von Daten außerhalb der SCORM Spezifikation müsste jeder Kurs eigene Funktionen bereitstellen oder LMS-fremde Dienste aufrufen. Class 3 dagegen erfordert aufgrund der massiven SCORM- und LMS-Veränderungen sehr viel Aufwand sowohl bei der Erfassung der nötigen SCORM-Erweiterungen als auch deren Umsetzung am LMS. Diese Erweiterungen wären auch schwer übertragbar auf andere LMS.

Die Wahl von Class 2 bedeutet geringe Erweiterung der SCORM API und des Datenmodells, und Bereitstellung von Funktionen außerhalb von SCORM, die z.B. durch Web Services nach dem Prinzip der Service Oriented Architecture17 bereitgestellt werden, auf die Simulationskurse zugreifen können. Beide Themen werden im nächsten Abschnitt vorgestellt.

4.3 Erweiterung von SCORM und SCORMSimuService

Hier werden die Erweiterungen von SCORM, die vorgenommen wurden, um die Simulation Game of Life in einen SCORM Kurs integrieren zu können, beschrieben. Dazu zählen eine neue API Funktion zum Erfragen der URLs der an dieses LMS angemeldeten externen Dienste, die dazu notwendige SCORM Datenmodellerweiterung sowie ein externer Dienst namens SCORMSimuService zur Verarbeitung und Speicherung der Daten von Game of Life.

API Erweiterung – Zentrale Verwaltung der URLs der externen Dienste

Es ist heutzutage üblich, verschiedene Funktionalitäten auf unterschiedliche Rechner zu verteilen. Sei es aus Ressourcengründen, oder weil diese Funktionen von anderen Parteien angeboten werden. Der Zugriff auf verteilte Dienste erfordert aber Kenntnis der jeweiligen URL. Damit nicht in jedem SCORM Kurs die URL eines benötigten externen Dienstes fest codiert werden muss, wurde die API um eine Funktion LMSGetServiceURL(ServiceName) erweitert. Über diese lassen sich die Addressen zu einem durch ServiceName identifizierten Dienst abrufen.

LMSGetServiceURL(ServiceName):

- Parameter ServiceName ist ID des benötigten Dienstes als String

- Rückgabewert die URL des Dienstes als String

Da die API üblicherweise als Zusammenspiel von JavaScript und Java Applet realisiert wird, kann die schon vorhandene JavaScript API leicht um diese Funktion erweitert werden, und die eigentliche Funktionalität dem Applet überlassen werden. In diesem Fall ruft das Applet eine URL auf, die einen festen Pfad relativ zur URL des LMS besitzt. Hinter dieser URL verbirgt sich ein Java Servlet, das dann den eigentlichen Rückgabewert liefert. Wie die URLs der registrierten Dienste auf einem LMS verwaltet werden, ist zunächst relativ unwichtig. Zu Testzwecken wurde hier lediglich eine Properties Datei mit Key-Value-Paaren angelegt, die die URLs zu jedem Dienst enthält.

SCORMSimuService

Generell sollte die Aufgabe eines externen Dienstes sein, Daten und Funktionen für einen Simulationskurs bereitzustellen, die ohne größere Erweiterung von SCORM nicht möglich sind. Die Aufgabe des SCORMSimuServices ist es, die im Kapitel 4.1 genannten Daten der Simulation zu speichern und abrufen zu lassen. Ziel war es nicht, umfangreiche Funktionalitäten bereitzustellen, sondern prinzipiell die Funktionsweise von Class 2 zu demonstrieren.

Der SCORMSimuService wird über eine URL mittels HTTP-GET oder HTTP-POST aufgerufen. Als Query können bzw. müssen gewisse Parameter übergeben werden. Das sind zum einen die Art der geforderten Operation, und die Eingabedaten der Operation. Die URL kann zuvor über die neue API Funktion erfragt werden.

Die verfügbaren Operationen sind:

1. put – speichert einen Datensatz zu einem Simulationslauf
2. get – liefert alle Simulationsläufe zu einem User
3. create – initialisiert die Datenbank

Hinter der URL verbirgt sich hier wieder ein Servlet, das die Querydaten interpretiert und anschließend die nötigen Datenbankoperationen ausführt. Als Datenbank wurde das leicht zu integrierende HSQLDB18 verwendet.

4.4 Zusammenwirken der Komponenten

Zu Demonstrationszwecken wurde ein Frameset erstellt, welches in Grundzügen dem des LMS smartBLU entspricht, und in dem die entwickelten Komponenten zusammen auf ihre Funktionsweise getestet werden können. Es gibt einen Frame, der die SCORM API (Javascript) und das API Applet enthält. Dieser ist normalerweise nicht sichtbar. Ein zweiter Frame, der LMS Frame, welcher zur Navigation im LMS dient. Und ein dritter, der Course Frame, welcher den gela-

denen SCORM Kurs enthält.

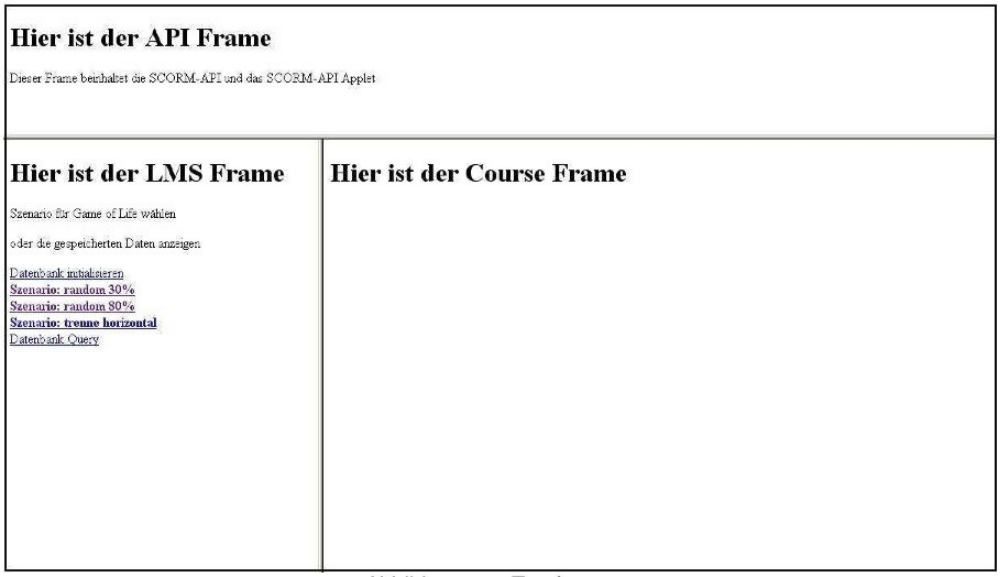

Abbildung 9: Frameset der Testapplikation

Der API Frame stellt die SCORM API bereit als JavaScript. Dazu muss eine Variable mit dem Namen API existieren. Jeder SCORM Kurs muss als erstes diese API finden, bevor er gestartet werden kann. Dieses Verfahren ist in SCORM fest vorgeschrieben. Über die API Variable können dann sämtliche SCORM API Befehle ausgeführt werden. Hier ist allerdings nur die API Erweiterung implementiert. Die Erstellung und Ausführung eines gesamten SCORM Kurses mit Nutzung des gesamten Spektrums an API Funktionen war nicht Ziel des Tests.

Mit der Wahl eines Szenarios im LMS Frame wird im Course Frame eine entsprechende Seite geladen, die die Game of Life Simulation als Applet enthält und diese mit den richtigen Parametern initialisiert. Das würde später Bestandteil eines kompletten SCORM Kurses sein. Die Simulation kann nun ausgeführt werden. Nach dem Beenden mittels „Finish“ kann sich der Kurs über die neue API Funktion (LMSGetServiceURL) die URL seines benötigten Dienstes holen. In diesem Fall LMSGetServiceURL(“GameOfLife“). Der Kurs wird nun per HTTP-Get eine Speicherungsaufforderung an den SCORMSimuService schicken und ist anschließend beendet.

Um sich die erfassten Daten anzeigen zu lassen, kann der Link „Datenbank Query“ benutzt werden. Es werden dann alle gespeicherten Simulationsergebnisse zu einem User angezeigt.

Die Folgende Abbildung zeigt die Interaktionen der Komponenten in einem Sequenzdiagramm am Beispiel eines Simulationslaufs. Die API Erweiterung „SCORMSimuService“ wurde nicht als eigenständiges Objekt definiert, da es im eigentlichen Sinne Bestandteil des LMS ist.

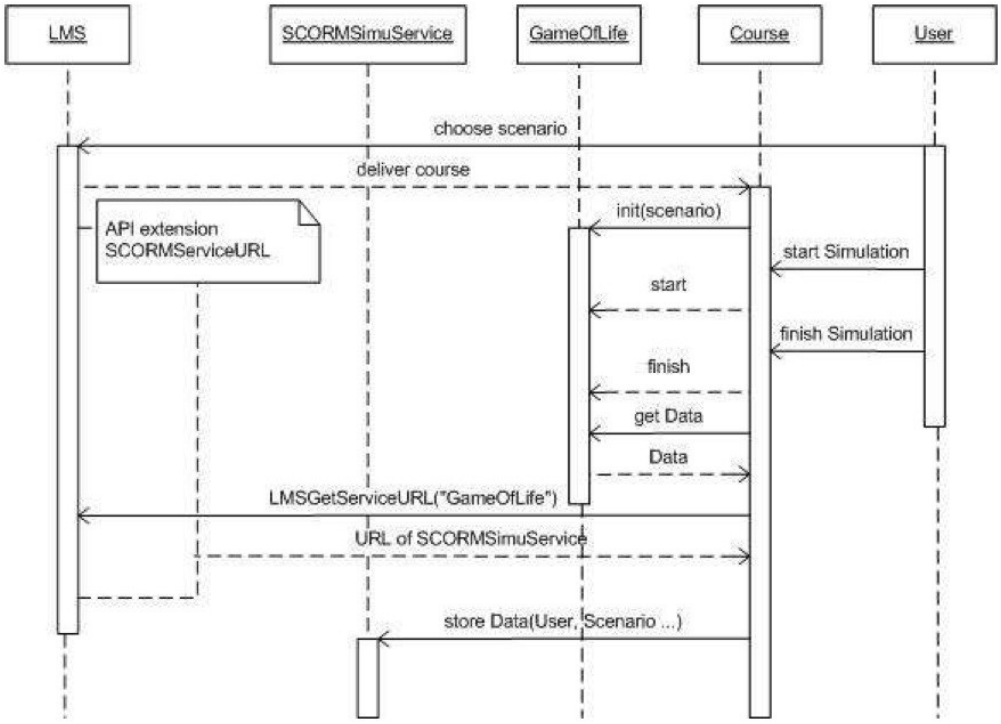

Abbildung 10: UML Sequenzdiagramm zum Ablauf eines Simulationslaufs

4.5 Ergebnis

Es folgt eine Übersicht der Ergebnisse der Implementierung. Eine ausführlichere Erklärung und Schlussfolgerungen erfolgen im Fazit.

1. Umsetzung des Class 2 Konzepts.
2. SCORM API Erweiterung LMSGetServiceURL(ServiceName).
3. Wahlfreie Verteilung der externen Dienste im Netz durch die API Erweiterung.
4. Erstellung des exemplarischen Dienstes SCORMSimuService zum Speichern von Simulationsdaten.
5. Erfolgreicher Test der API Erweiterung und des Dienstes durch die selbst erstellte Simulation Game of Life in einer LMS-ähnlichen Umgebung.

5 Fazit und Ausblick

Die Implementierung der Testumgebung zeigt, dass durch das Class 2 Konzept mit geringen SCORM Veränderungen Simulationen in Lernkurse integriert werden können. Die API Erweiterung zur Verwaltung der URLs von externen Diensten ist dabei nur eine Möglichkeit von vielen, trotz SCORM Inkompatibilität dem LMS eine gewisse Kontrolle über diese Dienste zu gewähren und den Zugriff auf diese zentral zu gestalten. Auf diese Art kann das LMS einen Pool an Diensten bereitstellen. Mittels geeigneter Metadaten könnten diese beschrieben werden, und SCORM Kurse gleichzeitig ihre Anforderungen an das LMS angeben. Falls dann ein benötigter Dienst fehlt, könnte dieser auch von Drittanbietern eingeholt werden auf diese verwiesen werden. Ein SCORM LMS könnte so sehr vielseitige Funktionen in Form einer Service Oriented Architcture anbieten.

Generell ist es relativ einfach, die SCORM API um einige Funktionen zu erweitern, solange diese nicht mit anderen SCORM Funktionen interferieren. Trotzdem sollte diese Vorgehensweise mit Vorsicht bedacht sein, wenn man auf Kompatibilität und Austauschbarkeit von Kursen mit anderen LMS bewahren möchte. Jede Änderung müsste auf anderen SCORM LMS ebenfalls vollzogen werden.

Was das Class 2 Konzept ebenfalls ermöglicht, ist eine Inter-SCO-Kommunikation. Ein externer Dienst kann z.B. die Synchronisation von Kurs-interner Kommunikation ermöglichen. Bisher bieten viele SCORM LMS zwar Kommunikationskanäle in Form von Chats oder Foren an, allerdings sind diese allgemein und nicht kursspezifisch. Lernkurse könnten gezielter die Kommunikation mit anderen Kursteilnehmern erleichtern, oder nach Beiträgen zu einem aufgetretenen Problem in Foren suchen. Wenn man noch mehr Aufwand zu leisten bereit ist, ließen sich auch Multi User Lernkurse auf diese Weise erstellen, was durch die SCORM Spezifikation nicht möglich ist.

Was die Möglichkeiten des Class 3 Konzepts angeht, so bleibt wahrscheinlich abzuwarten, ob die Forschungen seitens JADL und DMSO Ergebnisse bringen. In dieser Arbeit konnten dazu aus Zeitgründen keine Tests vorgenommen werden.

Ausblick

In welche Richtung eine SCORM Erweiterung gehen kann, zeigte bereits das Fazit. Generell könnten so beliebige Zusatzfunktionen für Lernkurse bereitgestellt werden. Weitergehende Tests sollten auf jeden Fall mit einem kompletten Lernkurs in einem SCORM LMS laufen. Zum Thema Webfähigkeit von Simulationen könnte man Game of Life als Thinclient gestalten, und lediglich die Visualisierung in den Lernkurs integrieren, und die ausgelagerte Logik als externen Dienst gestalten, um so Aussagen über Performance und Datenaufkommen treffen zu können. Jüngste Untersuchungen im Fraunhofer hatten gezeigt, dass ein Ansatz mit Web Services als Kommunikationsmittel nur schlecht skaliert. Das Thema

Thinclient scheint daher nicht trivial zu sein. Ein weiterer interessanter Punkt wären angepasste Metadaten für im Lernkurs eingesetzte Simulationen. Diese würden Simulationen auf ähnliche Weise beschreiben und bei der Auswahl helfen, wie es bereits bei allgemeinen SCORM Kursen funktioniert. Dazu zählen z.B. die Einordnung in definierte Typen (siehe Abschnitt 2.2.5) oder Aussagen zu geeigneten Kriterien. Neben inhaltlichen Werten können auch formale Aspekte beschrieben werden wie Class x Kompatibilität bzw. Erfordernis, Thin- / Fatclient und benötigte externe Funktionen.

Abbildungsverzeichnis

Literatur- und Verweisverzeichnis

Alle Links wurden zuletzt Oktober 2006 überprüft.

[STANGLE] Werner Stangle

eLearning, E-Learning, Blended Learning

http://www.stangl-taller.at/ARBEITSBLAETTER/LERNEN/Elearning.shtml

[MEDER] Norbert Meder

Didaktische Ontologien

http://www.bonn.iz-soz.de/wiss-org/beitraege/Meder.doc

[HAEFELE] Kornelia Maier-Haefele, Hartmut Haefele

Learning-, Content- und Learning-Content-Management-Systeme: Gemeinsamkeiten und Unterschiede

http://www.qualifizierung.com/download/files/LMS-CMS-LCMS.pdf

[MSO] Managerseminare Online

e-Lexikon

http://www.managerseminare.de/msemi/3546443/frontend/elexikon_liste.html?kat=2000

[ALLERT] Heidrun Allert

Instructional Roles in eLearning Metadaten Standards

http://projekte.l3s.uni- hannover.de/pub/bscw.cgi/d7151/Instructional%20Roles%20in%20eLearning%20Metadata%20Standards.%20pps-Slides.ppt

[SMITH] Brent Smith

SCORM-Simulation Interface Standards Study Group Status Report

http://www.jointadlcolab.org/newsandevents/ifests/2006/presentations/Mr_Brent_Smith.ppt

[AICC] AICC

Simulation-Based Instructional Systems Components and Topologies. An AICC Position Paper

http://www.sisostds.org/index.php?tg=fileman&idx=get&id=35&gr=Y&path=Position+Papers&file=AICC+Position+Paper.doc

[ADL/HLA1] DMSO, JADL

ADL/HLA Integration Research Blueprint

http://www.jointadlcolab.org/research/2004/sita/sita_research_blueprint.doc

[ADL/HLA2] Katherine L. Morse; Victor P. DiRienzo, Jr., Alion; Jake Borah

Simulation Assisted Learning Using HLA and ADL

http://www.movesinstitute.org/xmsf/events/JfcomWorkshopMay2003/HLA-ADL-overview-v3.ppt

[HALL] Brandon Hall

E-Learning online Glossary

http://www.brandon-hall.com/free_resources/glossary.shtml

[OUA] Open Universities Australia

http://www.open.edu.au/

[SOA] Australien-Info.de

Schools of the Air

http://www.australien-info.de/school-of-the-air.html

[PRENSKY] Marc Prensky

True Believers: Digital Game-Based Learning in the Military

aus "Digital Game-Based Learning (McGraw-Hill, 2001)" – Chapter 10

http://www.marcprensky.com/writing/Prensky%20-%20Digital%20Game-Based%20Learning-Ch10-Military.pdf

[SD] Rudolf Schröder, Dirk Wankelmann
Theoretische Fundierung einer e-Learning-Didaktik und der Qualifizierung von e-Tutoren
http://www.rudolf-schroeder.de/download/p-etutor-1d.pdf

[CAMPEN] Ralf von Petersdorf-Campen
Semantische Dienste in heterogenen Knowledge-Management- und eLearning-Umgebungen
http://dbis.informatik.uni-rostock.de/Studium/Diplomarbeiten/1098791620.76_0

[CEL91] F. Cellier: Continious System Modelling. Springer-Verlag. 1991

[BOS94] H. Bossel: Modeling and Simulation. Vieweg-Verlag, 1994

[ZPK00] B, Zeigler, H. Praehofer, T.G. Kim: Theory of Modeling and Simulation. Academic Press, 2000

[BCN00] J. Banks, J.C. Carson, B.L. Nelson: Discrete Event Simulation. Prentice Hall, 2000 (3. Auflage)

[BAU96] B. Baumgarten: Petri-Netze: Grundlagen und Anwendungen. Spektrum Verlag, Heidelberg, 1996

[MAR04] A. Martens: Ein Tutoring Prozess Modell für fallbasierte Intelligente Tutoring Systeme. Akademische Verlagsgesellschaft Aka GmbH, Berlin 2004, Dissertation

[KW78] KORN, G. A., J. V. Wait: Digital continuous-system simulation. Prentice-Hall, Englewood Cliffs, NJ, 1978

[ZEI84] Bernard Zeigler: Multifaceted Modeling and Discrete Event Simulation. Academic Press, London, 1984

[MOR61] George W. Morgenthaler: The theory and application of simulation and operations research, Progress in Operations Research 1. John Wiley and Sons, New York, 1961

Stichwortverzeichnis