Nutzungssimulator für adaptive Lernspiele und Computersimulationen

Bachelorarbeit von

Lukas Bach

KIT - Karlsruhe Institut für Technologie
Fraunhofer IOSB - Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung

Erstgutachter:	Prof. Dr.-Ing. J. Beyerer
Zweitgutachter:	Prof. Dr. rer. nat. H. Steusloff
Betreuender Mitarbeiter:	Dipl.-Inf. Alexander Streicher

15. Mai 2018 – 15. September 2018

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, 15. September 2018

. . . . . .

(Lukas Bach)

Abstract

Within the scope of this work a simulator for user behaviour, called *ElaiSim*, was developed. The simulator can be integrated into various applications from the E-Learning field. It is intended to be used to systematically test external Adaptivity Software, which improves the course of the game in Educational Serious Games.

This thesis describes the concept of storing and transmitting user behaviour in the context of E-Learning in a standardized way. Various existing serialization techniques are described which solve this problem. The protocol *xAPI* is of particular interest, as it is being used in the implemented ElaiSim application. Furthermore, it is described how such standardization techniques can be used to simulate user behaviour and thus systematically support the development of adaptive learning software.

From a technical point of view, the application focuses on the adaptivity software *E-Learning A.I.* (ELAI), which can be integrated into serious games to make adaptive changes to the course of the game in order to improve the learning experience. ElaiSim can mimic the actions performed by a real person within the serious game and intercept and verify the result values of the ELAI. The simulator also provides a graphical user interface for specifying and editing well-defined play scenarios and test cases, with which the results of the ELAI can be systematically verified.

The implementation is realized as a NodeJS application with a React based user interface. Various components have been developed as RESTful web services in order to allow an interoperable communication with the ELAI as well as other applications.

Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Simulator für Benutzerverhalten, genannt *ElaiSim*, entwickelt. Der Simulator kann in verschiedene Anwendungen aus dem E-Learning Bereich integriert werden und das Verhalten des Spielers nachahmen. So sollen externe Adaptivitätssysteme, die konstruktiv in den Spielverlauf von Educational Serious Games eingreifen, systematisch getestet werden.

Die Arbeit beschreibt konzeptionell, wie Benutzerverhalten im Kontext zu E-Learning auf standardisierte Art und Weise gespeichert und übertragen werden kann. Dazu wird auf verschiedene bestehende Serialisierungsmechanismen eingegangen, die dieses Ziel erfüllen. Insbesondere das Protokoll *xAPI* wird thematisiert, da es in der realisierten ElaiSim verwendet wird. Außerdem wird beschrieben, wie solche Standardisierungen verwendet werden können, um Benutzerverhalten zu simulieren und damit systematisch die Entwicklung von adaptiver Lernsoftware zu unterstützen.

Aus technischer Sicht liegt der Fokus auf der *E-Learning A.I.* (ELAI), einer in Serious Games integrierbaren Software, die adaptive Eingriffe zur Steigerung des Lerneffektes im Spielverlauf vornimmt. ElaiSim kann die Handlungen einer echten Person im Serious Game nachahmen und die Ergebniswerte der ELAI abfangen und verifizieren. Darüber hinaus liefert ElaiSim eine grafische Oberfläche zur Spezifizierung und Editierung von klar definierten Spielverläufen und Testfällen, mit denen die Ergebnisse der ELAI systematisch verifiziert werden können.

Die Implementierung ist als NodeJS Anwendung mit einer auf React basierenden Oberfläche realisiert. Verschiedene Komponenten sind als RESTful Webservices entwickelt worden, um eine interoperable Kommunikation mit der ELAI sowie weiteren Anwendungen zu ermöglichen.

Inhaltsverzeichnis

KAPITEL 1

Einführung

Im Laufe der Zeit sind in der Gesellschaft verschiedene Vorgehensweisen zur Optimierung des Lernprozesses entwickelt worden. Digital unterstütztes Lernen wurde in der Vergangenheit weitestgehend nur durch Kursplattformen zur Strukturierung von Lernmaterialien verbessert, die den gewöhnlichen Lernprozess ordnen. In der heutigen Zeit dagegen ergeben sich andere, neuere Lernverfahren. Ein aktueller Trend ist mit sogenannten Educational Serious Games das Konzept, Videospiele im E-Learning zu verwenden um effektivere Lernergebnisse zu erzielen.

Die entwickelten Serious Games verschmelzen dazu die Grundziele von sowohl Lernplattformen als auch Videospielen: Unterhaltung und Wissenserwerb. Ein Serious Game soll ähnlich wie sonstige Spieltitel den Spieler derartig fesseln, sodass er die Spielaufgaben freiwillig und mit Freude erfüllt. Gleichzeitig spielt der Lerneffekt eine bedeutende Rolle, da der Spieler aus dem Serious Game Wissen und Erfahrung mitnehmen soll, die er später in einem produktiven Umfeld anwenden können soll. [Dör+16a]

Um sowohl den Unterhaltungsfaktor als auch den Lerneffekt zu maximieren, werden die Serious Games mit *adaptiven Spielelementen* ausgestattet, die das Spielverhalten beobachten und den Spielverlauf abhängig davon adaptieren. So können zum Beispiel komplexere Aufgaben gestellt werden, falls der Spieler unterfordert wird, oder genauere Hilfestellungen geliefert werden, falls der Spieler die Spielaufgaben nicht lösen kann. [SS16]

1.1. Problemstellung

Die Problemstellung dieser Arbeit befasst sich mit der Entwicklung von adaptiven Lernsystemen. Das Adaptivitätssystem, welches auf Basis des Nutzerverhaltens Analysen durchführt um so den Verlauf der Lernsysteme konstruktiv zu beeinflussen, soll hierbei strukturiert und deterministisch getestet werden. So wird eine systematische Weiterentwicklung des Systems ermöglicht.

Auf der technischen Seite soll der Fokus auf dem bereits implementierten Adaptivitätssystem *E-Learning Artificial Intelligence* (ELAI) liegen, auf welches in Abschnitt 3.2 genauer eingegangen wird.

Konzeptionell ist zu klären, auf welche Arten Nutzerverhalten standardisiert und gespeichert werden kann. Die Fragestellung schließt mit ein, wie sich solche Verhaltensmodelle

zu Simulationszwecken nutzen lassen können und wie sich Simulationen von Verhaltensmustern manuell oder automatisch parametrisieren lassen.

Im Zuge des systematischen Testens des Adaptivitätssystems ist insbesondere eine programmatische Verifikation der Reaktionen diesen Systems von Interesse. So soll getestet werden, ob sich das System in Abhängigkeit des gemessenen Nutzerverhaltens korrekt verhält und korrekte Schlüsse zieht.

1.2. Lösungsansatz und Zielsetzung

Im Rahmen dieser Arbeit soll ein Nutzungssimulator implementiert werden, welcher über eine Weboberfläche steuerbar ist. Die Verhaltensmuster, welche durch den Simulator modelliert werden, sollen dem in Abschnitt 3.3 beschriebenen xAPI Protokoll und damit einem klar definierten Standard folgen. Alternative Protokolle und Modelle zur Standardisierung von Verhalten werden ebenfalls im Rahmen dieser Arbeit beleuchtet.

Der Nutzungssimulator soll hierbei insbesondere vier grundlegenden Funktionen ermöglichen, (1) die Aufnahme von Nutzerverhalten aus externen Lernsystemen, (2) das manuelle Bearbeiten der Verhaltensmuster in einem visuellen Editor, (3) das Abspielen der Verhaltensmuster in ein Adaptivitätssystem sowie (4) die Verifikation der Korrektheit dieses Systems durch eine dynamische Analyse seiner Adaptivitätsantworten. Somit soll der Simulator auch als Testumgebung für solche Systeme dienen können.

Für die Funktionen zum Aufnehmen und Abspielen des Nutzerverhaltens ist insbesondere die Standardisierung der Verhaltensmuster von Interesse. Aus diesem Grund befasst sich ein integraler Bestandteil der Arbeit mit den zunächst konzeptionellen und dann auch umgesetzten Standards und Protokollen, mit denen die verschiedenen Werkzeuge kommunizieren sollen.

1.3. Projektumfeld

Diese Arbeit ist im Kontext zum Projekt „Intelligente Bildauswertung in verteilten Systemen“ (IBIS) entstanden. An dieser forscht die Abteilung „Interoperabilität und Assistenzsysteme“ (IAS) des Fraunhofer Instituts für Optronik, Systemtechnik und Bildauswertung (IOSB) in Karlsruhe. In dem Projekt wird an E-Learning Lösungen gearbeitet, die durch die Realisierung von Interoperabilität und Adaptivität eine möglichst konstruktive und effektive Lernerfahrung bei dem Benutzer erreichen soll. [Str17]

Hierbei sind einige Educational Serious Games (zum Beispiel die in Abbildung 1.1 dargestellten Spiele SaFIRa und Lost Earth 2307) sowie die Software ELAI entstanden, welche an derartige Spiele angekoppelt werden kann, um durch adaptive Spieleingriffe den Lerneffekt und Spielspaß zu maximieren [Bie16; SR17a]. Die Architektur der ELAI basiert auf RESTful Webservices, sodass ohne großen Aufwand neue Serious Games oder andere Anwendungen angekoppelt werden können, solange sie die spezifizierten Protokolle verwenden. Daher kann eine wie in der Zielsetzung beschriebene Simulationsanwendung auf ähnliche Weise an die ELAI angeschlossen werden, um mit ihr zu kommunizieren.

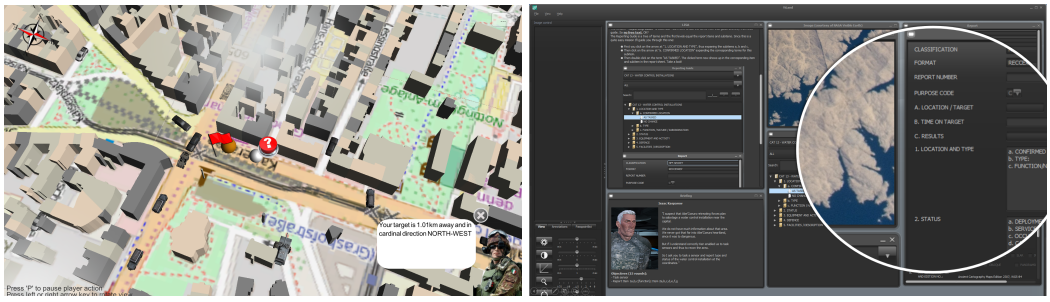


Abbildung 1.1: Bildschirmausschnitte aus den Lernspielen SaFIRa (siehe Abschnitt 3.1.3.2) und Lost Earth 2307 (siehe Abschnitt 3.1.3.1) [Bie16; SR17a]

1.4. Überblick über die Arbeit

Kapitel 2 liefert einen Überblick über den aktuellen Stand der Forschung und Technik. Hier wird auf bestehende Modelle und Konzepte zur Standardisierung von Nutzerverhalten sowie auf existierende Implementierungen zur Simulation von Nutzeverhalten eingegangen.

Kapitel 3 erklärt die in dieser Arbeit verwendeten Konzepte und Technologien. Dabei werden zunächst die grundlegenden Konzepte von E-Learning sowie die dazu verwendeten Technologien wie die E-Learning A.I. und die Experience API erklärt. Danach werden die Frameworks und Technologien beschrieben, die in moderner Webentwicklung Anwendung finden und auch im Rahmen dieser Arbeit verwendet wurden.

In Kapitel 4 wird ein Entwurf zur Standardisierung von Benutzerverhalten vorgestellt. Damit sollen die konzeptionellen Fragestellungen zur Standardisierung und Verifizierung beantwortet werden.

Die technische Implementierung von *ElaiSim*, welche die praktische Problemstellung der Arbeit lösen soll, ist in Kapitel 5 präsentiert. Hier wird auf den Lösungsansatz und die Architektur der entwickelten Anwendung eingegangen.

Die Arbeit endet mit einer technischen Evaluation in Kapitel 6, welche einen typischen Anwendungsfall der realisierten Implementierung beschreibt und diskutiert, sowie einem abschließenden Fazit in Kapitel 7.

KAPITEL 2

Stand der Forschung und Technik

In diesem Kapitel wird auf bestehende Forschungsarbeiten und Implementierungen im Kontext dieser Arbeit eingegangen. Hierzu wird zunächst ein Überblick über bestehende Möglichkeiten geliefert, wie Benutzerverhalten standardisiert werden kann. Da im praktischen Teil dieser Arbeit ein Simulator für Benutzerverhalten entwickelt worden ist, beschreibt der zweite Abschnitt dieses Kapitels ähnliche Projekte, in deren Rahmen solche Simulatoren entstanden sind.

2.1. Standardisierung von Nutzerverhalten

Da sich diese Arbeit mit der Modellierung von Nutzerverhalten befasst, ist es vor Allem interessant, auf welche Weise Nutzerverhalten standardisiert werden kann. Weil der Benutzer im Allgemeinen für eine Anwendung immer eine zentrale Rolle spielt, existiert für fast jeden Anwendungsfall ein spezialisiertes Standardisierungsverfahren für die Nutzerinteraktionen oder den Nutzer selbst.

2.1.1. Bestehende Konzepte der Verhaltensmodellierung

Im Folgenden werden verschiedene Arbeiten beschrieben, in deren Umfang eine Modellierung von Benutzerverhalten zur systematischen Problemlösung eingeführt und verwendet wurde.

Leng u.a. beschreiben, wie Verhaltensmodellierung und Verhaltensanalyse genutzt werden können, um Netzwerkqualität und die Qualität von sozialen Anwendungen zu verbessern [Len+16]. Dafür wird ein Nutzermodell definiert, welches Aspekte wie Surfgewohnheiten und demografische Daten nutzt. Auf ähnliche Weise ist durch Prangchumpol u.a. beschrieben, wie durch Analyse von Nutzerverhalten Serverkosten eingespart werden können [PST09]. Mithilfe von Assoziationsregeln über Verhaltensmuster kann Rechenleistung eingespart werden, da die Häufigkeit von Zugriffen durch Nutzer vorhergesagt werden kann. Temiyasathit u.a. legen dar, wie mittels der Analyse von Nutzerverhalten in einem E-Learning System Aktivitäten in einem Lernkurs vorhergesagt werden können [TPS16]. Dabei liegt der Fokus auf in der Regel wöchentlich stattfindenden Kursen. Im Gegensatz zu dieser Arbeit werden also eher selten auftretende Nutzeraktionen modelliert.

In der Arbeit von Xie u.a. ist ausgeführt, wie aus Verhaltensmustern die aktive Betrachtungsdauer von Videos in E-Learning Systemen vorhergesagt werden kann [Xie+17]. Dort sind Ereignisse wie Abspielen/Pausieren, Spulen oder Verlassen des Videos Grundlage für die Verhaltensmodellierung.

Wie sich zeigt, gibt es in vielen verschiedenen Anwendungsfällen das Potential, Verhaltensmodellierung zu verwenden. Die Art und Weise, wie Verhalten und Interaktionen einer Person modelliert werden und darauf basierend ein Benutzermodell erzeugt wird, variiert stark und ist dabei von dem Anwendungsfall abhängig. Jede Modellierung orientiert sich an einer Parametrisierung des Verhaltens, die für den Anwendungszweck notwendige Eigenschaften miteinbezieht.

In dieser Hinsicht hebt sich der Standard *ActivityStream* von den zuvor genannten Konzepten ab [SP17]. Hierbei handelt es sich um ein universelles Protokoll, mit dem Benutzeraktivität übertragen und verarbeitet werden kann. Aktivitäten werden mit dem *JavaScript Object Notion* (JSON) Format serialisiert und entsprechend einer vom *World Wide Web Consortium* (W3C) entwickelten Struktur formatiert. Damit ergibt sich ein universell einsetzbarer Standard, der für diese Arbeit interessant ist und dem später tatsächlich verwendeten Standard sehr ähnelt. Ein Beispiel für eine ActivityStream konforme Aktivität ist in Abbildung 2.1 beschrieben.

```
{
  "@context": {
    "@vocab": "https://www.w3.org/ns/activitystreams",
    "ext": "https://seuss-extension.example/terms/",
    "@language": "en"
  },
  "summary": "Food preference",
  "type": "Statement",
  "content": "I would not like green eggs and ham",
  "ext:vegetarian": false,
  "ext:preference": "bad"
}
```

Abbildung 2.1: Beispiel eines ActivityStream Objektes.

2.1.2. Verhaltensmodellierung im E-Learning Bereich

Die bisher genannten Anwendungsfälle stehen entweder nicht im Kontext zu E-Learning oder konzentrieren sich weitestgehend auf Kursmanagement und webbasierte Lernsysteme. Die für diese Arbeit im Vordergrund stehenden Educational Serious Games liefern als aktuelles Forschungsthema ebenfalls viele Möglichkeiten der Verhaltensmodellierung. Insbesondere dort, wo ein Soll-Verhalten durch das Spiel vorgegeben und vom Nutzer eingelernt werden soll, ist eine Betrachtung der Verhaltensmuster des Nutzers interessant. Im Gegensatz zu den in Abschnitt 2.1.1 vorgestellten Modellierungsverfahren unterscheiden sich diese weniger stark voneinander, da sie ähnliche Ziele verfolgen.

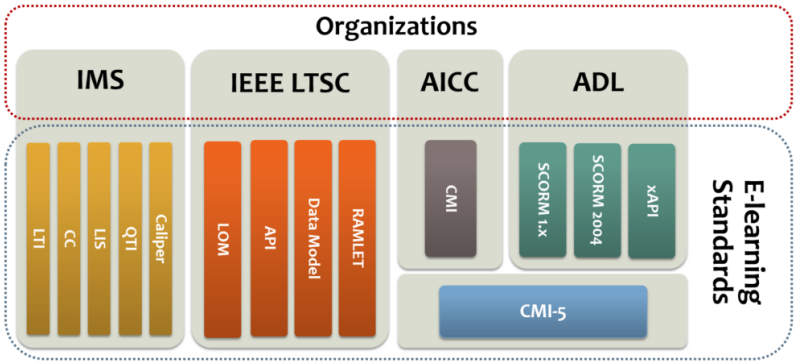


Abbildung 2.2: E-Learning Standards und Organisationen, von denen sie entwickelt wurden (Bildquelle [Bak+17])

Arnold u.a. zeigen in ihrer Arbeit auf, wie Adaption in Serious Games erreicht werden kann, indem das Spiel Informationen über den Spieler sammelt und daraus ein Nutzermodell generiert [Arn+13]. Das Modell basiert dabei weitestgehend auf den vom Benutzer aufgenommenen Handlungen. Das Spiel orientiert sich an einem vordefinierten nichtlinearen Handlungsstrang. Entsprechend dem Nutzermodell werden dabei Pfade aus der Handlung vorgegeben, um Adaption zu realisieren.

Berdun u.a. definieren ein weiteres Benutzer- und Verhaltensmodell, welches konkret an einem im Rahmen deren implementierten Serious Game orientiert ist [BA18]. Das Serious Game ist für mehrere Spieler ausgelegt und motiviert die Zusammenarbeit zwischen ihnen. Die Aktionen der Spieler werden mitverfolgt und für jeden wird ein Benutzerprofil anhand seiner Aktionen generiert. Dadurch können kollaborative Features aus jedem Spieler extrahiert werden, ausgehend von einem Benutzerprofil, seiner derzeitigen Aktion und dem aktuellen Spielzustand.

2.1.3. Standards für Verhaltensmodellierung im E-Learning Bereich

Die zuvor aufgeführten Werke verwenden keine interoperablen standardisierten Vorgehensweisen zur Modellierung von Person und Verhalten. Stattdessen werden jeweils spezielle Modellierungen definiert, die den individuellen Anforderungen der einzelnen Arbeiten gerecht werden. Es existieren für bestimmte Bereiche aber auch standardisierte Modelle und Protokolle. Hier ist die Arbeit von Bakhouyi u.a. hervorzuheben, in welcher die Standardisierung und Interoperabilität von verschiedenen E-Learning Systemen dokumentiert wurde [Bak+17]. Es werden vier große Organisationen sowie die durch diese entstandenen Standards und Empfehlungen beschrieben, die universell im E-Learning Bereich angewandt werden können und welche die Entwicklung von derartigen Standards vorangetrieben haben: Das *Global Learning Consortium* (IMS), das *IEEE Learning Technology Standards Commitee* (LTSC), das *Aviation Industry CBT Committee* (AICC) sowie die *Advanced Distributed Learning Initiative* (ADL). Die Organisationen und ihre entwickelten Standards sind in Abbildung 2.2 aufgeführt.

Die ersten drei genannten Organisationen, IMS, LTSC und AICC, haben jeweils Standards entwickelt, die in erster Linie der Strukturierung von digitalen Lernmaterialien dienen.

Es existieren dabei auch konkrete Protokolle um zum Beispiel die Aufgabenstellung und die Performance des Bearbeiters bei Online-Quizzen zu modellieren. Interoperabel anwendbare Standards im E-Learning Bereich, die über gewöhnliche kursorientierte Systeme hinausgehen und universelle Verhaltensmodellierung unterstützten, werden nur durch die zuletzt genannte Organisation ADL entwickelt. [Bak+17]

Der wichtigste durch das ADL entwickelte Standard hierzu ist das *Experience API* (xAPI) Protokoll, welches in der Implementierung dieser Arbeit auch Anwendung findet und ebenfalls von der in Abschnitt 3.2 erläuterten *E-Learning Artificial Intelligence* (ELAI) verwendet wird [KR16; Str17]. xAPI basiert wie auch ActivityStream auf dem Serialisierungsformat JSON. Durch xAPI werden Aktionen als komplexe Objekte mit individualisierbarer Parametrisierung beschrieben. Damit ist xAPI interoperabel einsetzbar und kann in verschiedenen Anwendungszwecken als Standard verwendet werden. Die Spezifikation von xAPI ist in Abschnitt 3.3 im Detail erklärt. Die Struktur und Vorgehensweise ist bei xAPI sehr ähnlich zu ActivityStream, der Fokus und die Motivation von Ersterem liegt aber stärker auf dem Kontext von E-Learning, weshalb xAPI für diese Arbeit von größerem Interesse ist.

2.2. Simulation von Nutzerverhalten

Über die Verhaltensstandardisierung hinaus ist vor allem die Simulation von Nutzerverhalten ein bedeutender Teil dieser Arbeit. Direkt im Kontext zum E-Learning wurde kein bestehendes Projekt gefunden, welches Nutzerverhalten von Lernenden auf eine Art und Weise modelliert und simuliert, sodass es für diese Arbeit interessant wäre.

Ein Projekt, das diesem Nahe kommt, ist das in der Arbeit von Dyck in 2011 beschriebene Verfahren, um eine Person im Kontext zum arbeitsbegleitenden Lernen zu modellieren [Dyc11]. Obwohl hier der Fokus nicht auf den Handlungen und dem Verhalten, sondern auf der Person selbst liegt, zeigen sich hier dennoch Parallelen in der Modellierung zu zuvor genannten Protokollen wie xAPI, da auch dieses den Nutzer als Teil der Handlungen modelliert. Die Arbeit konzentriert sich stark auf die Diskussion, welche Informationen für das Nutzermodell relevant und sinnvoll sind. Dabei werden auf viele verschiedene Arten von Informationen eingegangen, darunter Stamm- und technische Daten, die allgemein für die Person interessant sind, aber auch kontextbezogene Informationen wie Lerneinheit und -präferenzen oder Fähigkeiten. Auch die Umgebung des Nutzers wie die verwendete Software und Geräte, Lokalisation und Umweltbedingung werden erörtert.

Damit ergibt sich ein sehr ausführliches Modell, welches im Kontext von E-Learning Sinn ergibt. Aus technischer Sicht ist das Benutzermodell als XML-Schema realisiert. Es wurde außerdem ein Simulator entwickelt, der den Kompetenzlevel einer Person in Abhängigkeit seiner (Lern-)Parameter berechnet. Der Simulator unterscheidet sich insofern von dem Ziel dieser Arbeit, indem er nur eine statische Repräsentation der Person zu einem gewissen Zeitpunkt analysiert und daraus Schlüsse zieht, während hier der zeitliche Aspekt und der Ablauf von Handlungen im Vordergrund stehen. Ein Bildschirmausschnitt des Simulators ist in Abbildung 2.3 zu sehen.

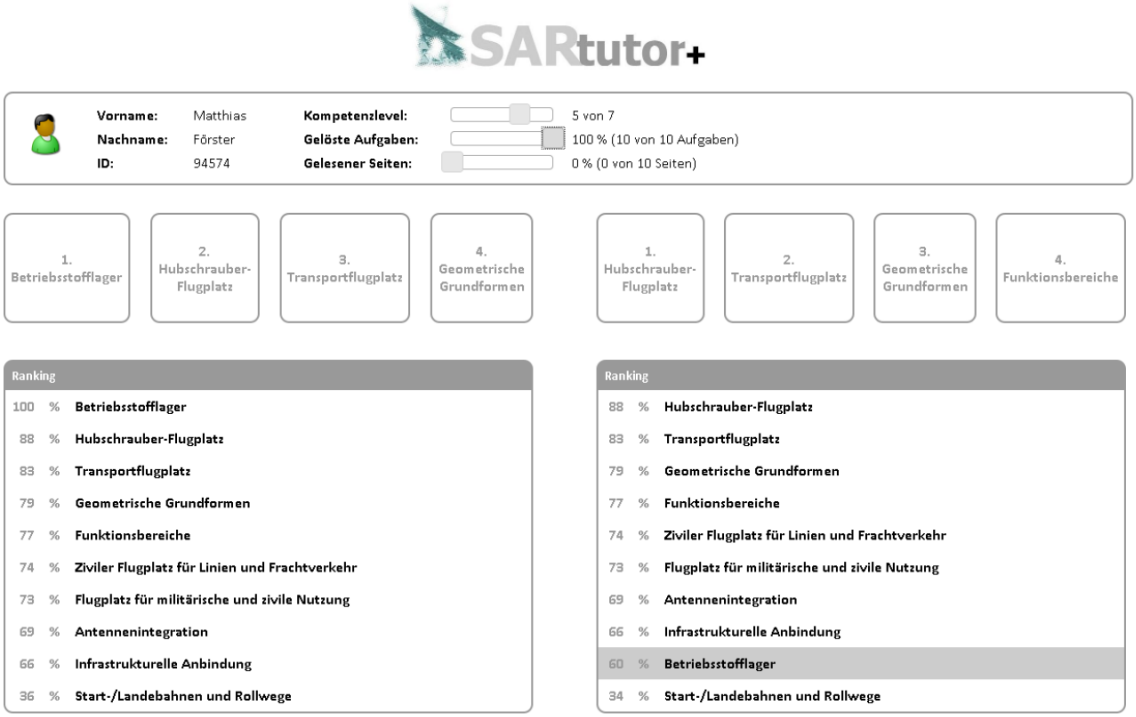


Abbildung 2.3: Der von Dyck implementierte Nutzersimulator. Parameter wie „Gelöste Aufgaben“ und „Gelesene Seiten“ beeinflussten den Kompetenzlevel. (Bildquelle [Dyc11])

KAPITEL 3

Grundlegende Konzepte

In diesem Kapitel wird auf die verschiedenen Konzepte und Technologien eingegangen, die sowohl für den theoretischen Teil der Arbeit als auch für die technische Realisierung relevant sind. Zunächst wird auf die Thematik des E-Learning sowie die im Kontext dazu entstandenen Technologien eingegangen. Diese sind für die Motivation der Arbeit sowie für den in Kapitel 4 vorgestellten Entwurf zur Standardisierung von Benutzerverhalten wichtig.

Danach werden verschiedene Frameworks vorgestellt, für die weniger E-Learning im Vordergrund steht, sondern die stattdessen für die allgemeine Entwicklung von Webanwendungen und die später vorgestellte Implementierung interessant sind.

3.1. Adaptives E-Learning und Serious Games

In der heutigen Zeit wird der Lehrbereich wie auch viele andere Bereiche durch elektronische Hilfsmittel unterstützt. Nicht selten werden Online-Quizze zur Selbstevaluation oder Einstufung der Lernenden, Kurs Management Systeme oder ähnliches genutzt. [Hsi16]

Um den Lernprozess aktiv zu motivieren, ist die Verwendung von Videospielen zu Lernzwecken ein aktuelles Forschungsthema. Hier ist das Ziel, den Unterhaltungszweck von Videospielen zu einem euphorischeren und fesselnderen Lerneffekt zu nutzen.

In den folgenden Abschnitten wird zunächst das Konzept von Videospielen zu Lehrzwecken erläutert. Dann wird darauf eingegangen, wie digital unterstütztes Lernen durch adaptive Hilfsmittel verbessert werden kann. Beides stellt die Motivation zum Anwendungszweck dieser Arbeit dar.

3.1.1. Serious Games und Gamification

Im modernen Bereich des E-Learning hat sich der Begriff „Serious Game“ als Überbegriff für Videospiele gebildet, in denen spielerisch eine reale Situation nachgestellt wird, aus der der Spieler einen konstruktiven Nutzen ziehen kann. In dieser Arbeit werden *Educational Serious Games* als Teilmenge der Serious Games thematisiert, welche konkret den Lerneffekt in einem bestimmten Gebiet als Ziel setzen. Daher wird im folgenden der Begriff „Serious Game“ abkürzend mit der Bedeutung von „Educational Serious Games“ verwendet.

In derartigen Spielen wird eine reale Situation simuliert, in der die Lösung einer zu trainierenden Aufgabe erfordert wird. Es werden spielerische Konzepte angewandt, die auch aus üblichen Videospielen bekannt sind, allerdings steht das Lösen der Aufgabe und der daraus resultierende Wissenserwerb im Vordergrund. Dadurch wird erreicht, dass der Spieler sich mit ihm zuvor fremden Szenarien vertraut macht, mit denen er in der realen Welt umgehen können muss, gleichzeitig kann aber auch bereits erlerntes Wissen vertieft und trainiert werden. [JT17]

Beispiele für solche Videospiele sind Brandsimulationen, in denen die Brände möglichst effektiv gelöscht werden sollen oder Flugsimulationen, in denen verschiedene Manöver geübt werden sollen. [Sti17]

Der Begriff der Serious Games wurde erstmals durch Clark Abt definiert [Abt87]. Er hat den Nutzen von Abstraktionen als Notwendigkeit für einen effizienten Lernprozess beschrieben und erläutert, wie die Übertragung von Lernszenarien auf Spiele genutzt werden kann, um effektiver zu lernen.

Neben Serious Games hat sich ebenfalls der Begriff „Gamification“ etabliert. Gamification beschreibt ähnlich wie Serious Games die Integration von Entertainment in industrielle und wirtschaftliche Prozesse, um mehrwertschöpfende Prozesse zu ergänzen. Die genaue Abgrenzung zwischen beiden Begriffen ist unscharf definiert und weicht je nach Quelle ab. Den häufigsten Definitionen zufolge sind (Educational) Serious Games vollständige Videospiele, die einen Lernbereich komplett abdecken und als selbstständiges Lernmedium dienen, während Gamification nur vereinzelte spielerische Elemente nutzt, um Produktivität zu erhöhen. [Sti17]

3.1.2. Steigerung des Lernprozesses durch Adaptivität

Ein Serious Game verfolgt bei dem Spieler zwei essentielle Ziele: Einen Lerneffekt zu erzielen und dabei Wissen und Fähigkeiten zu vermitteln, sowie den Spieler ausreichend zu vergnügen und zu motivieren, sodass der Lerneffekt nicht als mühselige oder vergebliche Aufgabe wahrgenommen wird. Entwurf und Design des Spielablaufes und die dabei aufkommenden Aufgabenstellungen tragen dazu besonders bei. Dennoch kann trotz gutem Spielentwurf der Spielverlauf für den Nutzer als repetitiv oder redundant erscheinen, wenn der Spieler mit vielen Grundlagen des zu erlernenden Wissens vertraut ist oder bereits Übung darin gesammelt hat. Auf der anderen Seite kann ein Spieler sehr schnell durch die Aufgabenstellungen überfordert werden, wenn diese auf erfahrenere Spieler ausgelegt sind oder die Geschwindigkeit der Entwicklung des Spielverlaufes sich nicht an der Lerngeschwindigkeit des Spielers orientiert. Beides mindert sowohl den Lerneffekt als auch die Motivation des Spielers. Hier wird das Konzept der Adaptivität interessant: Ein adaptives Spiel reagiert auf den Spieler und steuert den Spielfluss in Abhängigkeit von seinem Spielverhalten und seinem Hintergrund, indem das Spiel kontextabhängig modifiziert wird. Solche Modifikationen können das im Spiel zu lösende Problem vereinfachen, indem beispielsweise Abkürzungen ermöglicht werden. Genauso kann das Problem erschwert werden, indem behindernde Spielelemente eingebracht werden und so von dem Spieler mehr Aufwand zum Erreichen des Ziels eingefordert wird. Auslöser für solche Modifikationen können basierend auf dem Spielerprofils definiert werden, wie zum Beispiel die Anzahl der gespielten Stunden des Spiels oder der Klassifizierung der

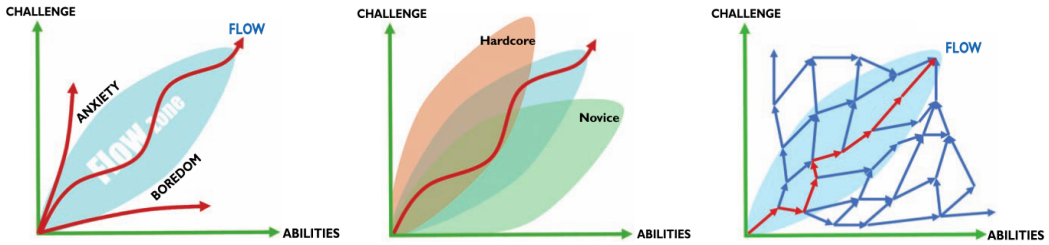
Person (wie „Schüler“, „Studierender“, „Doktorand“), oder auch aufgrund intelligenter Analysen des Spielverhaltens, wie zum Beispiel der Detektion von häufig sich wiederholenden identischen Aktionen des Spielers. Letzteres kann darauf hinweisen, dass der Spieler sich nicht mehr genau über seine Aufgabe bewusst ist und adaptive Unterstützung hilfreich sein kann. [SS16]

3.1.2.1. Flow

Ein für Adaptivität interessantes Konzept ist das des *Flows*. Flow beschreibt einen psychologischen Zustand, in dem eine Person sich hochkonzentriert und fokussiert einer Aktivität widmet und dabei Spaß und Vervollständigung empfindet. Während sich eine Person in diesem Zustand befindet, kann sie ihre gesamte Aufmerksamkeit auf eine einzige Aufgabe richten, ohne dem Risiko von Ablenkungen oder Abschweifungen zu unterliegen. [Che07]

Dieses Prinzip wurde ursprünglich durch den Psychologen Mihaly Csikszentmihalyi definiert [Csi88]. In der Arbeit wird vor allem auf den psychologischen Hintergrund dieses Bewusstseinszustandes eingegangen. Csikszentmihalyi beschreibt, dass der menschliche Verstand sich für gewöhnlich in einem ungeordneten, chaotischen Zustand befindet, was Konzentration und Aufmerksamkeit auf konkrete Aufgaben erschwert. Wird der Flow-Zustand erreicht, beginnt der Verstand sich zu ordnen und Gedanken, Gefühle und Sinne fokussieren sich alle auf dasselbe Ziel. Zufolge der Arbeit kann dieser Zustand über viele verschiedene Arten erreicht werden, einschließlich äußerer Stimuli, die den Verstand von außen heraus ordnen.

Dabei lassen sich leicht Parallelen zu der Vorgehensweise und den Zielen der Adaptivität von Serious Games erkennen, da beide dasselbe Ziel verfolgen. Adaptivität versucht einen ebensolchen Bewusstseinszustand herbeizuführen, in dem der Spieler seinen gesamten Fokus auf das Spiel legt, ohne das Gefühl von Einschränkung durch zu einfache oder Überforderung durch zu komplexe Aufgabenstellungen zu empfinden. Dies wird durch die beschriebenen äußeren Stimuli erreicht, welche im Falle der Adaptivität die Anpassungen des Spielverlaufs an die Bedürfnisse des Spielers sind.



(a) Einzelne Spieler weisen sehr individuelle Spielverhalten auf.

(b) Daher werden Spieler durch Adaptivitätsmechanismen eingestuft.

(c) Adativität sorgt für einen individuell angepassten Spielverlauf.

Abbildung 3.1: Die Darstellung von Flow und ihren Einfluss auf die Adaptivität in Spielen (Bildquelle [SZ07])

Visuell wird Flow im Kontext der Serious Games oft als Diagramm dargestellt, welches aus den Achsen „Herausforderung durch das Spiel“ und „Fähigkeiten des Spielers“ besteht

(Abbildung 3.1). Der Spielverlauf eines konkreten Spielers lässt sich auf diesem Diagramm als Linie darstellen, die vom Ursprung in einem ausgeglichenen Gewicht zwischen Herausforderung und Fähigkeiten verläuft. Diese Linie verläuft durch den „Flow-Bereich“, ein Idealbereich, in dem der Spieler im Flow-Bewusstseinszustand gehalten wird. Befindet er sich unterhalb des Bereiches, wird er unterfordert, da seine Fähigkeiten die Herausforderung übertreffen. Darüber wird er überfordert, da das Gegenteil der Fall ist (Abbildung 3.1a). Da verschiedene Spieler verschiedenen Profilen folgen, hat jeder einen anderen Idealbereich, weshalb ein Spiel mit guter Realisierung der Adapivität den Spieler entsprechend seiner Fähigkeiten klassifiziert und den korrekten Flow-Bereich für ihn erkennt (Abbildung 3.1b). Das Spiel passt dann den Spielverlauf an das Spielerprofil an, um ihn so im Flow-Bereich zu halten und seine Konzentration und Effizienz zu maximieren (Abbildung 3.1c).

3.1.2.2. Adaptiver Zyklus

Eine mögliche Vorgehensweise, um Adaptivität in eine Lernanwendung zu integrieren, ist der von Shute und Zapata-Rivera definierte „Four-Process Adaptive Cycle“. Dieser Adaptive Zyklus beschreibt eine iterative Vorgehensweise, um adaptiv den Lernprozess an den Lernenden anzupassen und dabei den Nutzen zu maximieren. Der Ablauf ist in Abbildung 3.2 visualisiert und besteht aus den im Folgenden aufgelisteten vier Phasen. [SZ07]

Erfassung (Capture): Informationen über den Lernenden werden gesammelt, während er mit der Umgebung interagiert.

Analyse (Analyze): Die gesammelten Informationen werden verwendet, um ein domänenspezifisches Modell des Lernenden zu erzeugen.

Selektion (Select): Aus dem Modell werden relevante Informationen ausgewählt, die für die Realisierung der Adaptivität interessant sind.

Darstellung (Present): Die selektierten Informationen werden verwendet, um dem Lernenden kontextbezogene und aufbereitete Inhalte zu präsentieren.

Nachdem die vier Phasen durchlaufen sind, beginnt der Zyklus erneut, da der Lernende in Folge der Präsentation der neuen Inhalte sein Lernverhalten anpasst und das System darauf wiederum reagieren kann.

Dieser Zyklus beschreibt eine von vielen Möglichkeiten, Adaptivität in Lernszenarien zu erreichen. Die Vorgehensweise ist aber meist ähnlich: Der Lernende wird bei seinen Aktivitäten beobachtet, die Beobachtungen werden analysiert und darauf basierend wird der Lernprozess konstruktiv manipuliert.

Für diese Arbeit ist das Konzept der Adaptivität interessant, da sie Anlass zum Beobachten des Benutzers und dem Serialisieren seiner Verhaltensmuster liefert. Da sich diese Arbeit mit der Standardisierung von Nutzerverhalten befasst, ist Adaptivität in Lernspielen eine bedeutende Motivation und stellt auch den Rahmen der später vorgestellten Implementierung dar, welche in Kapitel 5 erläutert wird.

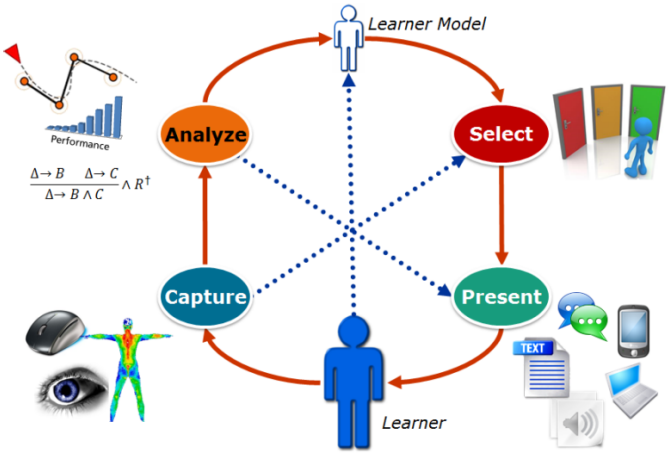


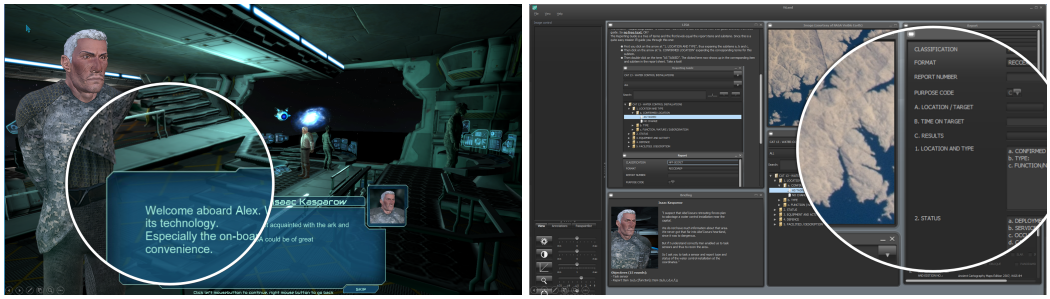
Abbildung 3.2: Ablauf der vier Phasen des Adaptiven Zyklus (Bildquelle [SS16], basierend auf [SZ07])

3.1.3. Beispiele für Serious Games

Im Rahmen des Projektes „Intelligente Bildauswertung in verteilten Systemen“ am Fraunhofer IOSB sind bereits einige Serious Games entstanden, die Adaptivität auf genau diese Vorgehensweisen realisiert haben. Zu diesen Spielen gehören auch „SaFIRa“ und „Lost Earth 2307“, die in dieser Arbeit oft als Referenzbeispiele erwähnt und daher im Folgenden kurz erläutert werden.

3.1.3.1. Lost Earth 2307

Lost Earth ist ein rundenbasiertes E-Learning Videospiel, in dem der Lernende Aufgaben in einer futuristischen Weltraumflotte übernimmt. Zu diesen Aufgaben gehört Ressourcenmanagement und die Lösung von Bildauswertungsproblemen. Das Spiel soll den Prozess der Bildauswertung trainieren und fachliche Kenntnisse übermitteln. Als sehr komplexes Spiel mit vielen Aufgabenbereichen ist es ein gutes Beispiel für produktiv eingesetzte Serious Games. [SRB17]



(a) Der Benutzer bekommt Aufgaben in einer spielerischen Umgebung.

(b) Im Spielverlauf müssen Bildauswertungsaufgaben gelöst werden.

Abbildung 3.3: Bildschirmausschnitte aus dem Lernspiel Lost Earth 2307 [SR17a]

3.1.3.2. SaFIRa

SaFIR (*Seek and Find for Image Reconnaissance*) ist als Seek-and-Find Spiel entwickelt und fordert den Spieler auf, ein Objekt auf einer geografischen Karte zu suchen. Dabei sieht der Spieler die Karte aus der Vogelperspektive und kann seine Figur durch Anklicken auf Zielpositionen bewegen. Bei den gesuchten Objekten handelt es sich um Panzerkampfwagen. Der Spieler muss den korrekten Panzer, der zu Beginn des Spiels gezeigt wurde, finden und seine Figur zu diesem bewegen. Dabei soll der Spieler darin trainiert werden, verschiedene Arten von Panzern zu unterscheiden.

Eine Erweiterung des Spiels ist SaFIRa (*SaFIR plus adaptivity*), welches eine adaptive Tutoreinheit zum Spiel ergänzt. Dabei handelt es sich um einen virtuellen Charakter, der innerhalb des Spiels dargestellt wird und über eine externe Anbindung zu einer Tutor AI (der in Abschnitt 3.2 beschriebenen ELAI) gesteuert wird. Auf Anfrage des Spielers liefert er Hilfestellung über die Richtung und Distanz zum gesuchten Zielobjekt. Weitere Ergänzungen durch Adaptivität sind verschiedene Anpassungen der Schwierigkeit in Abhängigkeit des Spielerverhaltens wie Größenanpassung des Zielobjektes oder Häufigkeit von überdeckenden Wolken und ablenkenden Objekten. Ein Bildschirmausschnitt mit eingeblendeter Tutoreinheit ist in Abbildung 3.4 zu sehen. [SRB17; Bie16]

Abbildung 3.4: Bildschirmausschnitt aus dem Lernspiel SaFIRa [Bie16]. Links unten ist die adaptive Tutoreinheit eingeblendet, welche dem Spieler Hilfestellung liefert.

3.2. E-Learning A.I. (ELAI)

Ein im Projektumfeld dieser Arbeit am Fraunhofer IOSB entstandenes Framework zur Realisierung von Adaptivität ist die Software „E-Learning A.I.“ (ELAI). ELAI ist als eigenständige Software entwickelt, die über externe Schnittstellen mit Serious Games kommuniziert.

Beide zuvor erwähnten Spiele, Lost Earth 2307 und SaFIRa, verwenden intern die ELAI, um adaptive Lernmethoden zu nutzen. [SR17a]

Die ELAI wird über einen an das Spiel individuell angepassten Adapter gekoppelt. Dieser Adapter wird direkt in das Spiel integriert und ist dafür zuständig, das Benutzerverhalten zu beobachten und an die ELAI zu senden sowie den Spielverlauf entsprechend der Ergebnisse der ELAI zu adaptieren. Die ELAI selbst ist dabei als externes Werkzeug dafür zuständig, die aus dem Serious Game empfangenen Daten zu verarbeiten und die Ergebnisse dem Adapter zu präsentieren.

Die Architektur und grundlegende Funktionsweise der ELAI ist durch Abbildung 3.5 dargestellt. In das Serious Game wird die Adapterkomponente („ELAI Interface“) eingeschleust, welche über eine Kommunikationsebene („Communication Layer“) mit der ELAI Daten austauscht. Der Adapter ist auf das Serious Game abgestimmt und muss konkret für ein spezielles Spiel entwickelt werden. Dadurch, dass der Adapter die spielspezifische Logik übernimmt, kann die ELAI selbst vielseitig auf verschiedene Arten von Spielen angewandt werden.

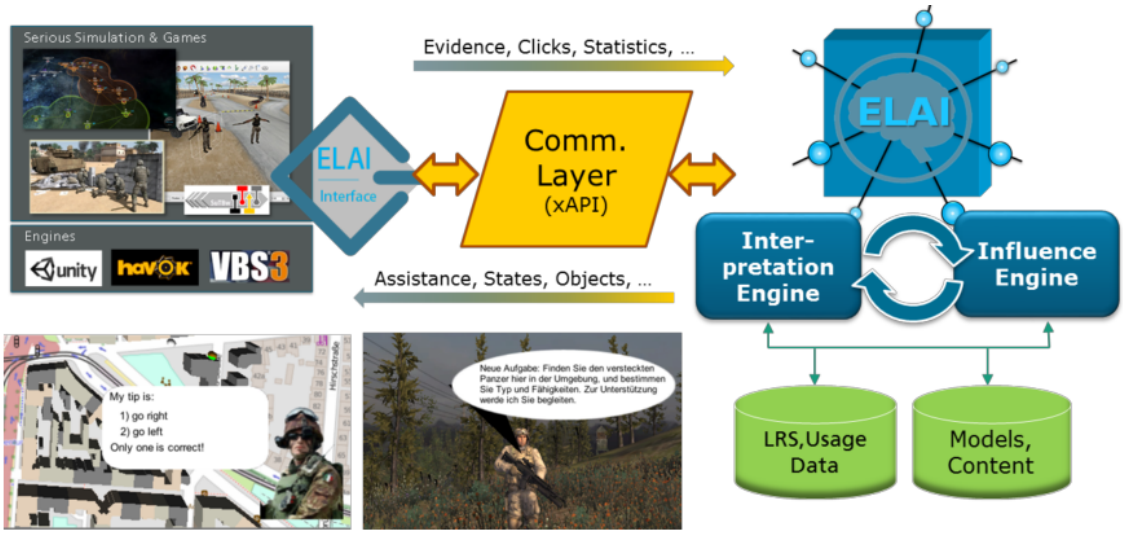


Abbildung 3.5: Architektur der ELAI und dem verbundenen Serious Game. ELAI kommuniziert über den Adapter („ELAI Interface“) mit dem Serious Game (Bildquelle [SRB17])

.

Die ELAI realisiert damit den zuvor beschriebenen vierphasigen Adaptiven Zyklus von Shute und Zapata-Rivera [SZ07]. Der Zusammenhang zwischen den Phasen des Zyklus und der ELAI ist im Folgenden dargestellt [Str17].

Erfassungsphase: Die ELAI wird mit dem Adapter verbunden. Der Adapter zeichnet Nutzerinteraktionen (wie Mausklicks, Tastatureingaben oder Spiel-spezifische Aktionen) auf und sendet diese an die ELAI.

Analysephase: Die ELAI extrahiert aus den Daten höherwertige Informationen wie „liest gerne Text“ oder „macht viele Wiederholungen“. Bereits in dieser Phase wird das gespeicherte Modell des Lerners angepasst.

Selektionsphase: Entsprechend vordefinierten Regeln wird von dem Nutzerzustand Empfehlungen für Anpassungen im Serious Game abgeleitet.

Darstellungsphase: Das Serious Game modifiziert entsprechend der Empfehlungen den Spielverlauf durch Adaption. Zum Beispiel können Hinweise als Hilfestellungen angezeigt werden, oder die Aufgabenstellung kann erschwert werden um sich dem Lernenden anzupassen.

Der durch die Phasen beschriebene Ablauf durchläuft so auch die Komponenten der ELAI. Dies ist in Abbildung 3.6 visualisiert.

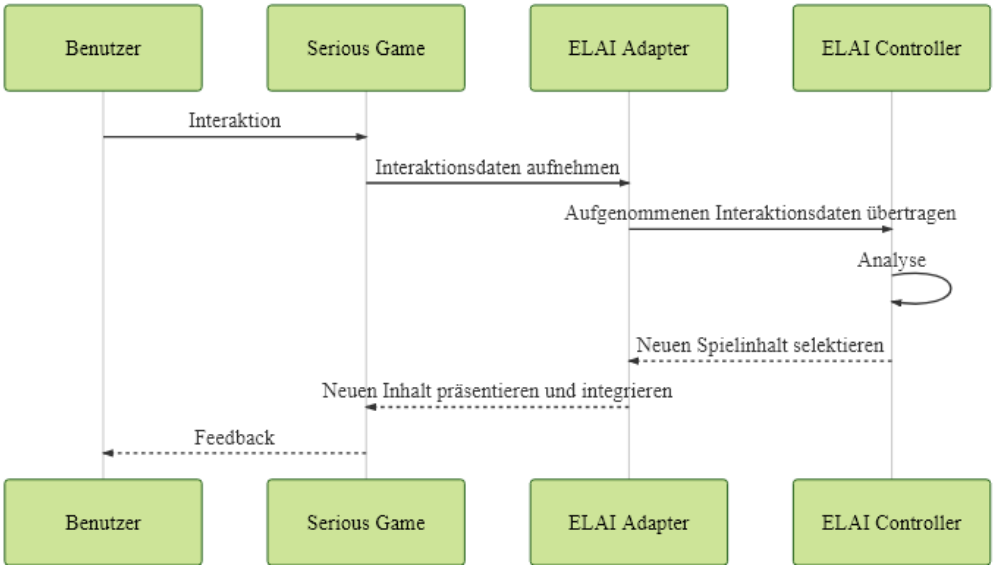


Abbildung 3.6: Der grundlegende Ablauf von Interaktionen eines Benutzers mit einem Serious Game und daran angeschlossene ELAI [Str17]

.

Die Kommunikation zwischen dem ELAI-Adapter und der ELAI selbst geschieht über HTTP-Nachrichten mithilfe eines standardisierten Protokolls. Dieses Protokoll basiert auf Objekten nach dem Format „JavaScript Object Notation“ (JSON) und verwendet zur Serialisierung und Übertragung des beobachteten Benutzerverhaltens das JSON-Schema der „Experience API“ (xAPI), welches in Kapitel 3.3 genauer erklärt wird.

Wie in Abbildung 3.5 dargestellt, besteht die ELAI aus zwei wichtigen Komponenten, der *Interpretation Engine* und der *Influence Engine*. Beide unterstützen verschiedene Phasen des adaptiven Zyklus. Die in der Erfassungsphase aufgenommenen Daten werden in der Analysephase durch die Interpretation Engine verarbeitet und in durch Maschinen interpretierbare Informationen umgewandelt. Dazu werden verschiedene Methoden aus der intelligenten Datenanalyse eingesetzt wie Clustering und Collaborative Filtering. In der Selektionsphase ist die Influence-Engine dafür zuständig, unter festgelegten Adaptionsstrategien zu wählen, um den Spielverlauf zu beeinflussen. Diese Strategien sind spezifisch für das jeweilige Serious Game definiert. [SR17a]

3.3. Experience API (xAPI)

In den vorigen Abschnitten sind verschiedene Anwendungsfälle vorgestellt worden, in denen Benutzerverhalten programmatisch verarbeitet wird, um Lernspiele dem Lernenden entsprechend zu adaptieren. Dafür ist eine standardisierte Serialisierung der Verhaltensmuster notwendig. Die zuvor vorgestellte Adaptivitätssoftware ELAI greift dabei auf den Standard *Experience API* (*xAPI*, zuvor als *Tin Can API* bekannt) zurück. Hierbei handelt es sich um ein auf dem *JSON*-Format (*JavaScript Object Notation*) basierenden Standard, welches Nutzeraktionen als komplexe Aktionen beschreibt. [KR16]

3.3.1. Vorgänger der Experience API

Die xAPI stammt von dem *Sharable Content Object Reference Model* (SCORM) ab, einer im Jahr 2000 entwickelten Spezifikation für E-Learning Systeme. SCORM beschäftigt sich dabei weitestgehend mit der Strukturierung von Lerninhalten und statischen Inhalten und verwendet dafür wiederum andere Protokolle. [Sof18]

Ab Version 2004 existiert in SCORM bereits die Möglichkeit, eine sequentielle Abfolge von Aktivitäten auf den Lernmaterialien zu definieren, die von den Nutzern verfolgt werden sollen. Damit kann bereits ein sequentieller Ablauf von Benutzeraktionen festgelegt werden, sodass Nutzerverhalten standardisiert werden kann. [Adv09]

Später im Jahr 2010 begann die Entwicklung der Tin Can API, aus der sich in 2015 die xAPI entwickelt hat.

3.3.2. Spezifikation der Experience API

Die xAPI ermöglicht es, Nutzeraktionen nach dem Format „Akteur Verb Object“ (zum Beispiel „John liest Moby Dick“) zu beschreiben. Solche Sätze werden nach den Definitionen von xAPI als *Activity Statements*, oder einfach Statements bezeichnet [KR16]. Ein Statement ist eine Datenstruktur mit mindestens drei Unterobjekten:

- Der *Akteur* (im Original *Actor*) beschreibt den Initiator der Aktion und damit das Subjekt des Statements. Hierbei kann es sich um eine einzelne Person handeln (bezeichnet als „Agent“) oder um eine Gruppe von Personen (Eine Gruppe von Agenten).
- Das *Verb* beschreibt, welcher Art von Aktion der Akteur nachgeht. Zu dem Verb ist in der Regel ein globaler Identifikationsschlüssel und eine menschenverständliche Beschreibung angegeben.
- Das *Objekt* (im Original *Object*) verknüpft die Aktion mit einer Instanz, die mit dieser zusammenhängt.

Neben diesen Parametern sind weitere Eigenschaften als optionale Parameter möglich, wie zum Beispiel ein Kontextobjekt, Zeitstempel oder Anhänge. Für den Kontext dieser Arbeit ist vor allem auch das *Ergebnis* (engl. *result*) von Interesse, welches als optionaler Parameter das Resultat der Aktion genauer beschreibt. Dieses Objekt gibt an, ob die Aktion

erfolgreich war sowie den Fortschritt der Aktion, falls es sich dabei um eine Teilhandlung handelt oder Ähnliches. Das zuvor vorgestellte Serious Game SaFIRa beispielsweise sendet sporadisch Statements mit der Distanz zum Ziel, um den Fortschritt und die Performance des Spielers messbar zu machen. [Adv18]

Ein Beispiel für ein xAPI Statement ist in Abbildung 3.7 dargestellt. Hierbei handelt es sich um ein Ausschnitt eines Statements, welches von SaFIRa gesendet wird, um den Spielfortschritt zu signalisieren. Der Akteur ist in diesem Fall der Spieler, welcher über seinen Namen und seine E-Mail Adresse identifiziert wird. Das Verb beschreibt die „Gespielt"-Aktion, die ausgeführt wird. In dem Objekt wird die Aufgabe spezifiziert, die der Spieler lösen muss. In dem zusätzlichen Ergebnis-Objekt wird der aktuelle aus dem Spiel inferierte Punktestand des Spielers sowie zusätzlich der Entfernungswert zum Ziel angegeben. [Bie16]

```
{
  "actor": {
    "objectType": "Agent",
    "name": "John Doe",
    "mbox": "mailto:john@doe.com"
  },
  "verb": {
    "id": "http://iosb.fraunhofer.de/.../played/",
    "display": {
      "en-US": "played"
    }
  },
  "object": {
    "objectType": "Activity",
    "id": "http://iosb.fraunhofer.de/.../findTheTank/",
    "definition": {
      "name": {
        "en-US": "Find the tank"
      }
    }
  },
  "result": {
    "score": {
      "scaled": 0.75
    },
    "extensions": {
      "http://iosb.fraunhofer.de/.../distanceToTarget": 0.77
    }
  },
}
```

Abbildung 3.7: Ausschnitt eines vom Serious Game SaFIRa erzeugten xAPI-Statements.

Alle in dem Statement verwendeten Instanzen, die über verschiedene Spielverläufe verwendet werden können, werden über sogenannte *Internationalized Resource Identifiers* (IRIs) identifiziert. Hierbei handelt es sich um eine Art Internet-Adresse, die (nicht notwendigerweise) auflösbar sein kann. Über IRIs werden unter anderem das Verb, das Objekt

und in diesem Fall die individuelle Eigenschaft „distanceToTarget“, welche im Ergebnis die Zieldistanz angibt, spezifiziert. Der Akteur dagegen wird über einen *Inversen Funktionalen Identifizierer* (*Inverse Functional Identifier*) eindeutig erkennbar gemacht. Hierbei handelt es sich um einen von vier möglichen Schlüsseln, die der Akteur definieren muss. Die möglichen Schlüssel sind in Tabelle 3.1 aufgelistet. [Adv18]

mbox	Die IRI der E-Mail Adresse des Akteurs.
mbox_sha1sum	Die hex-kodierte SHA1-Summe der IRI der E-Mail Adresse des Akteurs.
openid	Der über das Authentifizierungssystem *OpenID* nutzbarer Identifizierer des Akteuren.
account	Ein beliebig strukturiertes komplexes Objekt, welches den Benutzeraccount in einem anderen System beschreibt.

Tabelle 3.1: Verschiedene Möglichkeiten, einen Akteur in einem xAPI Statement über ein Inversen Funktionalen Identifizierer eindeutig identifizierbar zu machen. Quelle: [Adv18]

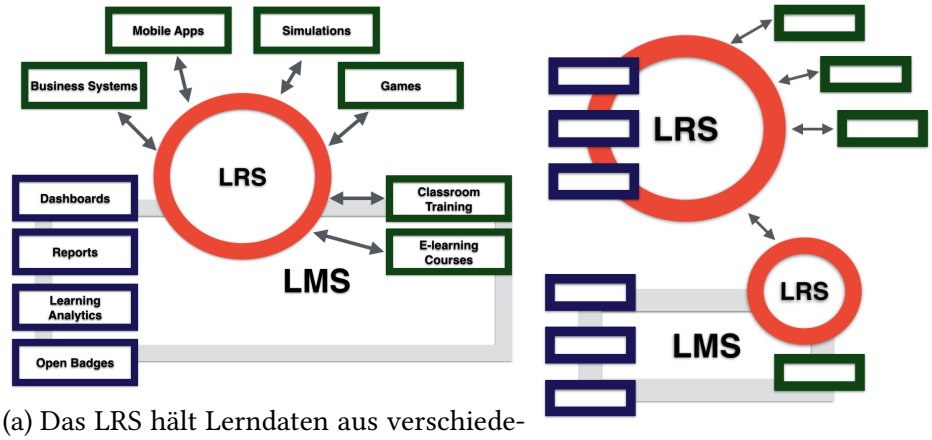
3.3.3. Interoperabilität mithilfe der Experience API

xAPI wurde nicht nur definiert, um Benutzeraktionen standardisiert abzuspeichern, sondern vor allem auch um diese zwischen verschiedenen Systemen auszutauschen. Schließlich liegt gerade in der durch xAPI gegebenen Interoperabilität der Vorteil, dass verschiedene unterschiedlich funktionierende Systeme sich dadurch auf denselben Standard verlassen und trotzdem die Daten einheitlich verarbeiten und verstehen können. Dadurch kann man zusätzliche Systeme an die Lernumgebung ankoppeln, die zum Beispiel Statistiken führen oder Analysen machen können. Für die ELAI existiert beispielsweise eine derartig angekoppelte Anwendung, die es über ein Web-Interface ermöglicht, verschiedene Adaptivitätsparameter wie Schwierigkeit manuell einzustellen. [SR17b]

Die Kommunikation findet mittels HTTP-Nachrichten statt. Durch die simple Kodierung mit JSON können einzelne Statements ohne großen Aufwand oder Komplexität für HTTP-Anfragen serialisiert werden. Die externen Systeme können dann als RESTful Web-Services eine Schnittstelle öffnen, über der Statements empfangen oder gesendet werden. In Kapitel 3.4 ist die Funktionsweise von RESTful Web-Services und HTTP-Nachrichten genauer erläutert.

Im Kontext der xAPI sind außerdem *Learning Record Stores* (LRS) definiert. Hierbei handelt es sich um einen Speicher, der Lerndaten in Form von xAPI Statements empfängt, speichert und auf Abfrage zur Verfügung stellt. Abbildung 3.8 skizziert die Umgebung von LRS Systemen, die mit *Lernplattformen* (*Learning Management System*, LMS) verknüpft sind. Das LRS empfängt Lerndaten in Form von xAPI Statements von verschiedenen Quellen wie zum Beispiel Serious Games und kann an Lernplattformen gekoppelt werden, welche Daten aus dem LRS holen und diese in verschiedenen Applikationen verwenden können. [Str17]

Wie zuvor erwähnt verwendet die ELAI das xAPI Protokoll, um das Nutzerverhalten aus dem Serious Game zu empfangen und zu verarbeiten. Der ELAI Adapter, der Spielspezifisch in das Serious Game eingebaut wird, beobachtet den Spieler und generiert



(a) Das LRS hält Lerndaten aus verschiedenen Anwendungen und kann in ein LMS integriert werden.

(b) Verschiedene LRS können miteinander verknüpft werden.

Abbildung 3.8: Skizze der Funktionsweise von Learning Record Stores (Bildquelle [Rus18]).

passende xAPI Statements, die an das ELAI Backend gesendet werden. Die ELAI hält diese Statements dann in einem dedizierten LRS, während in einer separaten Datenbank das Modell des Lernenden gehalten wird.

Die Implementierung des Nutzerverhalten-Simulators, der später in dieser Arbeit präsentiert wird, verwendet ebenfalls das xAPI Protokoll, um die Verhaltensmuster zu erzeugen und zu versenden. Dadurch kann der Simulator in die Umgebung der bestehenden Serious Games und der ELAI integriert werden, ohne diese anpassen zu müssen.

3.4. RESTful Web-Services

In der Vergangenheit wurden Webanwendungen häufig als statische Webseiten durch Serveranwendungen erzeugt und bei dynamischen Änderungen komplett neu übertragen. Mittlerweile bietet sich durch das Prinzip des *Representational State Transfe* (REST) ein Programmierparadigma, welches die Struktur von dynamischen Webanwendungen erheblich verbessert. Statt einen Webserver zu haben, der für sämtliche Logik einschließlich der Erzeugung der grafischen Benutzeroberfläche zuständig ist, kommuniziert die Oberfläche als eigenständige Anwendung mit Servern über sogenannte REST-Schnittstellen. Solche Schnittstellen definieren klar strukturierte Kommunikationswege, die auf HTTP-Nachrichten basieren. [Fie00]

Server, die solche Kommunikation unterstützen, werden als *RESTful Webservice* bezeichnet. In manchen Anwendungsfällen ist dabei eine grafische Benutzeroberfläche nicht notwendig, es können verschiedene Server einfach ihre gesamte Kommunikation über REST-Schnittstellen durchführen.

Eine Kommunikation über REST-Schnittstellen findet im Rahmen dieser Arbeit an zwei Stellen statt: Zum einen ist die in Abschnitt 5 beschriebene Implementierung, die in dieser Arbeit realisiert wurde, selbst in eine Weboberfläche und ein Server Backend getrennt, die beide über REST-Schnittstellen kommunizieren. Des Weiteren läuft auch die Kommunikation der in Abschnitt 3.2 beschriebenen ELAI weitestgehend nach dem

REST-Prinzip: Der ELAI-Adapter, der konkret für ein Serious Game entwickelt wird und in dieses integriert wird, sendet Nachrichten an die REST-Schnittstelle der ELAI und ruft über diese auch Performance-Daten über den Spielenden ab. Da die später vorgestellte Implementierung sowohl mit Serious Games als auch mit der ELAI über genau diese Schnittstellen kommunizieren muss, ist für die Implementierung entsprechend ein RESTful-Webservice realisiert worden, der diese Kommunikation ermöglicht.

3.5. Frameworks für moderne Webentwicklung

In diesem Abschnitt werden verschiedene Frameworks vorgestellt, die für die später vorgestellte Implementierung relevant sind. Die beschriebenen Werkzeuge und Technologien entsprechen dem aktuellen Stand der Technik bezüglich moderner Webentwicklung und werden auch bei anderen bekannten Projekten verwendet.

3.5.1. JavaScript

JavaScript ist eine dynamisch typisierte interpretierte Programmiersprache, die bereits 1995 von Netscape entwickelt wurde. Als Werkzeug, um Webanwendungen in Browsern den Massen zugänglich zu machen, hat sie große Beachtung und Verwendung gefunden. Da sie mittlerweile in allen gängigen Webbrowsern (und darüber hinaus auch in vielen anderen Anwendungen) unterstützt wird, ist sie beinahe unabdinglich für Webentwicklung sowie die Entwicklung von universell einsetzbare Anwendungen geworden. Parallel zu JavaScript existiert die Bezeichnung *ECMAScript*, dem Sprachstandard von JavaScript, der im Auftrag von Netscape durch die *European Computer Manufacturers Association* (*ECMA*) entwickelt wurde. Dieser Standard wurde seitdem aktiv weiterentwickelt und um viele verschiedene Konstrukte bereichert, sodass die Sprache eine Vielzahl von Paradigmen unterstützt. Damit werden objektorientierte, funktionale, prozedurale und andere Programmierstile ermöglicht. Zu diesem Standard existieren verschiedene Implementierungen von JavaScript Interpretern, die auch als *Engines* bekannt sind. Einige Webbrowser verwenden unterschiedliche Interpreter, welche den Sprachstandard teils mit geringfügigen Variationen unterstützen. Da vor allem nicht alle Browser die aktuellste Version von ECMAScript unterstützen, existieren entwicklungsunterstützende Werkzeuge, die Programmcode mit modernen ECMAScript Strukturen in umgewandelten Code konvertieren, der auch durch ältere Interpreter unterstützt wird. Dieser Prozess nennt sich *Transpilierung* (Abwandlung des Begriffes „Kompilierung“, da hier von einer Hochsprache zu einer anderen umgewandelt wird). [Fla11]

Die Sprache wurde ursprünglich für Webbrowser entwickelt, um Webseiten um dynamische Logik zu ergänzen. Aus diesem Grund existiert in der Grundform von JavaScript eine Standardbibliothek zum Manipulieren der angezeigten Webseitenstruktur sowie um Funktionen des Webbrowsers nutzen zu können. Solche Funktionen sind zum Beispiel das Speichern von geringen Datenmengen auf dem Benutzerrechner (Cookies), Senden und Empfangen von Nachrichten an den Server, der die Webseite bereitstellt, oder das Abrufen von Ereignissen wie Tastaturanschläge oder Mausklicks. Dabei ist zu erwähnen, dass JavaScript-Programme im allgemeinen asynchron ablaufen. Das bedeutet, dass beim

Aufrufen der Webseite ein initialer Programmcode in JavaScript kurz durchläuft und sämtliche weitere Methoden asynchron mit verschiedenen Ereignissen verknüpft und erst bei deren Aktivierung ausgeführt werden. Die Laufzeitumgebung ist dabei dafür zuständig, sämtliche Ereignisse zu handhaben und verknüpfte JavaScript Routinen entsprechend zu starten. Ein Beispiel für solche Asynchronität ist das Abrufen von Daten des Servers: Der JavaScript Code sendet über die Standardbibliothek eine HTTP-Nachricht an den Server, um Daten abzufragen. Dabei wird eine weitere Methode als Rückruffunktion (engl. *Callback*) registriert. Der Interpreter nimmt den Methodenaufruf auf der Bibliothek entgegen, sendet die Nachricht und wartet auf das Ergebnis. Sobald dieses eingetroffen ist, ruft der Interpreter die registrierte Rückruffunktion mit dem Ergebnis als Parameter auf. Dadurch entfällt für den JavaScript Code die Notwendigkeit, Multi-Threading zu betreiben oder durch aktives Warten Prozessorzeit zu verschwenden. Abbildung 3.9 illustriert diesen Vorgang. [Fla11]

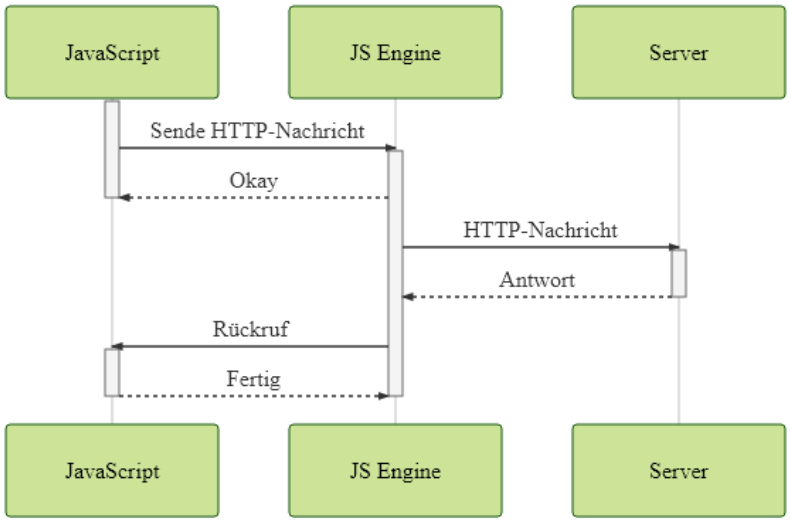


Abbildung 3.9: Beispiel für einen asynchronen Aufruf von Serverdaten durch ein JavaScript Programm.

3.5.2. Node.js und NPM

Neben der Browserumgebung, in der JavaScript für gewöhnlich läuft, haben sich mit der Zeit aber auch andere Umgebungen für verschiedene Anwendungszwecke entwickelt. Die Bedeutendste ist Node.js, eine JavaScript Laufzeitumgebung die 2009 mit dem Zweck entwickelt wurde, JavaScript Programmcode nativ außerhalb des Browsers laufen zu lassen. Ein großer Teil der JavaScript Standardbibliothek, der für die Manipulation der betrachteten Webseite zuständig ist, fällt hierbei aufgrund des abweichenden Anwendungsfalles weg. Dafür kommen einige Erweiterungen durch die Node.js Bibliothek dazu wie Zugriffe auf das Dateisystem oder das Starten von externen Prozessen. Damit ist es möglich, auch gewöhnliche Anwendungen zu entwickeln. Node.js basiert auf einer abgewandelten Form von Googles JavaScript Interpreter *V8*, welcher in Google Chrome verwendet wird. [TV10]

Node.js ist nicht zuletzt wegen der überdurchschnittlich großen Vielfalt an Paketen beliebt. Hierbei wird der *Node Package Manager* (*NPM*) verwendet, welcher direkt mit Node.js gebündelt ist. Zum Zeitpunkt dieses Schreibens existieren auf NPM über 750 Tausend veröffentlichte Pakete, die über die Kommandozeile installiert werden können.[1] Damit ist NPM das größte Repository von Paketen verglichen mit anderen Programmiersprachen.

Für Node.js Programme existieren verschiedene typische Anwendungsfälle. Node.js wird zum einen häufig für Entwicklungswerkzeuge verwendet, welche die Programmierung mit JavaScript unterstützen. Transpilierungsprogramme (welche zum Beispiel Programmcode nach modernem ECMAScript Standard in abwärtskompatiblen Code konvertieren) oder Testumgebungen für JavaScript Programme basieren oft auf Node.js. Außerdem ist es möglich, Webserver mit Node.js zu entwickeln. Dazu wird häufig die Bibliothek *express* verwendet (welche mit rund vier Millionen wöchentlichen Downloads zu den am häufigsten genutzten Bibliotheken auf NPM gehört[2]), ein Framework welches die Entwicklung von RESTful Webservices mit minimalem Programmierungsaufwand ermöglicht. In der später vorgestellten Implementierung wird express ebenfalls verwendet.

3.5.3. Performance von JavaScript und Node.js

Obwohl in Node.js wie in allen anderen JavaScript Laufzeitumgebungen der Programmcode auch hier interpretiert und nicht kompiliert wird und außerdem nur dynamisch typisiert ist, entsteht dabei ein Mehraufwand an Prozessorleistung bei der Ausführung. Da die Sprache außerdem Multi-Threading (die Ausführung des Programmes auf mehreren Prozessorkernen gleichzeitig) kaum unterstützt[3], wird JavaScript oft als ineffiziente und langsame Sprache angenommen. Durch die Asynchronität und die Verwendung von Rückruffunktionen zeichnet sich JavaScript tatsächlich aber als sehr effizient für konkrete Anwendungsfälle aus. Wenn ein Node.js Programm eine Datei lädt, kann es die Kontrolle über den Prozessor komplett abgeben und sich auf die Laufzeitumgebung verlassen, dass es die Kontrolle zurück erhält, sobald Daten eingetroffen sind. Das ist nicht nur effizienter als eigene Programmlogik zum Laden der Datei mittels Parallelisierung zu entwickeln, sondern birgt auch erheblich weniger Aufwand und Fehlerrisiko. Derselbe Vorteil ergibt sich bei dem Abarbeiten von Webanfragen, die normalerweise noch langwieriger ablaufen als Dateianfragen. Damit zeigen sich JavaScript Anwendungen sowohl ressourceneffizient als auch von geringem Entwicklungsaufwand für I/O-intensive Aufgaben. Webserver müssen meist nur HTTP-Anfragen steuern, Dateien laden und simple Berechnungen durchführen und sind somit eher I/O-intensiv als CPU-intensiv. Damit eignet sich Node.js hier dementsprechend gut als Laufzeitumgebung. [TV10]

[1] `https://npmjs.com`

[2] `https://npmjs.com/package/express`

[3] Tatsächlich existieren in JavaScript sogenannte Web Worker, die es ermöglichen, Programmcode in einem Hintergrundprozess ablaufen zu lassen. Dennoch ist JavaScript in erster Linie auf Anwendungen ausgelegt, in denen nur ein einziger Prozess läuft.

3.5.4. TypeScript

Ein Kritikpunkt von JavaScript ist die schwache dynamische Typisierung, die dem Entwickler zwar sehr viele Freiheiten erlaubt, aber auch eine Vielzahl an Software-Fehlern ermöglicht, die erst zur Laufzeit erkannt werden. Während die dynamische Typisierung schnelle Entwicklung von Software-Prototypen ermöglicht, ist aufgrund der Verbreitung von JavaScript in Webbrowsern die Verwendung in umfangreichen Produktivprojekten häufig notwendig. Daher sind verschiedene Projekte entstanden, die statische Typisierung ermöglichen: Der Google Closure Compiler[4], welcher JavaScript auf statische Typen überprüft und optimiert, Facebook Flow[5], eine JavaScript Spracherweiterung, die statische Typen in der Syntax ergänzt, und TypeScript[6], eine von Microsoft entwickelte Programmiersprache, die an JavaScript angelehnt ist, aber verschiedene syntaktische Konstrukte ergänzt. Zu diesen gehört nicht nur statische Typisierung, sondern auch Klassen[7], Interfaces und Enumerations [Mic18]. Damit motiviert TypeScript bessere Softwarequalität und verhindert das Auftreten von vielen Softwarefehlern bereits zur Kompilierzeit. [GBB17]

TypeScript lässt sich mit einem von Microsoft entwickelten Werkzeug zu JavaScript transpilieren. Hierbei kann auch eine ältere Version von ECMAScript als Transpilierungsziel gewählt werden, um Abwärtskompatibilität mit älteren Browsern zu erreichen. Bei der Transpilation werden sämtliche angegebene Typen überprüft und dann verworfen. Dadurch erhält man zwar nicht die Performancevorteile, die sonst durch starke statische Typisierung einhergeht, aber trotzdem die dadurch gegebene Softwarequalität.

3.5.5. React und Redux

Zwei moderne und beliebte Frameworks zur strukturierten Entwicklung moderner Webanwendungen sind *React* und *Redux*. React[8] ist eine von Facebook entwickelte Bibliothek zur Realisierung von grafischen Oberflächen. Sie ermöglicht die Aufteilung einer Anwendung in gekapselte austauschbare UI-Komponenten, die jeweils über eigene Programmlogik und einen internen Zustand verfügen.

Für jede Komponente kann Programmlogik zur Darstellung des Inhaltes abhängig vom Zustand und der Parametrisierung der Komponente festgelegt werden. Diese kann darüber hinaus weitere Komponenten als Kindelemente enthalten. So kann die Webanwendung als Baumstruktur von Komponenten angeordnet werden, während die Programmlogik auf diesen verteilt ist.

Da bei umfangreichen Anwendungen die komplette Verteilung der Programmlogik und des Programmzustandes auf einzelne grafische Komponenten aufwendig werden kann, ermöglicht die Bibliothek Redux die Definition eines globalen Programmzustandes mit klar strukturierten Zustandsübergängen. Hierzu müssen für eine Anwendung die folgenden Einheiten definiert werden:

[4] `https://developers.google.com/closure/compiler`

[5] `https://flowtype.org`

[6] `https://typescriptlang.org`

[7] Die aktuelle Version von ECMAScript definiert bereits auch einen Standard für Klassen, allerdings wurde dieser erst nach TypeScript eingeführt und unterstützt in der aktuellen Version Features wie private Klassenattribute noch nicht.

[8] `https://reactjs.org`

- Die Struktur des Anwendungszustandes
- Die möglichen Aktionen einschließlich ihrer Parameter, die durch jede Komponente auf dem Zustand durchgeführt werden kann
- Eine Reduzierungslogik auf dem Zustand, die eine alte Zustandsstruktur und eine durchzuführende parametrisierte Aktion auf einen neuen Zustand reduziert.

Der globale Anwendungszustand ist für alle Komponenten lesbar zugänglich. Um Änderungen am Zustand durchzuführen, muss eine der definierten Aktionen abgesendet werden. Wenn eine Redux-Aktion gestartet wird, leitet die Reduzierungslogik den neuen Zustand ab und alle UI-Komponenten, die Daten aus dem Zustand beziehen, werden neu erzeugt. Hierdurch kann die globale Anwendungslogik und der globale Zustand von lokalen visuellen Komponenten getrennt werden, was eine saubere und strukturierte Implementierung ermöglicht.

3.5.6. Distributierung mit Electron

Bei Software ist eine übliche Problematik die Interoperabilität auf verschiedenen Systemen. Eine Anwendung soll möglichst unabhängig von Betriebssystem, vorinstallierten Bibliotheken oder sonstigen Spezialanforderungen lauffähig sein. Für die Entwicklung von auf JavaScript basierenden Programmen bietet sich hier das Framework *Electron*[9] an. Dieses wurde von GitHub als Grundlage für ihren Texteditor Atom entwickelt und wird nun als Open-Source-Lösung von vielen bekannten Programmen wie Skype und Slack zur Erzeugung von interoperablen Anwendungspaketen verwendet.

Mit Electron kann eine JavaScript Anwendung zu einem portablen Paket mit ausführbarer Programmdatei gebündelt werden. Dabei wird die Anwendung mit dem NodeJS-Kern sowie *Chromium*, dem Browserkern von Google Chrome, verknüpft. Damit kann die Anwendung in einem nativen Fenster angezeigt werden ohne von sonstigen vorinstallierten Bibliotheken abhängig zu sein. Darüber hinaus kann das Anwendungspaket für alle gängigen Betriebssysteme erzeugt werden. Electron liefert über die JavaScript-Schnittstelle verschiedene universelle Operationen, um Betriebssystem-spezifische Funktionen wie das Erzeugen eines neuen Fensters zu ermöglichen.

[9] `https://electronjs.org`

KAPITEL 4

Standardisierung von Nutzerverhalten

In diesem Kapitel wird eine konzeptionelle Vorgehensweise zur Standardisierung von Benutzerverhalten vorgestellt. Das präsentierte Konzept ist im Kontext zu adaptivem E-Learning entwickelt worden, sodass besonders auf die dazu notwendigen Komponenten Wert gelegt wurde.

Da die Implementierung des Simulators zum einen eine interne Datenstruktur benötigt, um Verhaltensmuster abzuspeichern, zum anderen auch mit externer Software über ein standardisiertes Protokoll kommunizieren muss, ist eine konzeptionelle Standardisierung notwendig. Diese wird in Abschnitt 4.1 definiert und orientiert sich an der in Kapitel 3.3 eingeführten Spezifikation *Experience API* (xAPI), welche von der bestehenden Software im Projektumfeld dieser Arbeit bereits verwendet wird und auch von der in Kapitel 5 beschriebenen Implementierung genutzt wird. Damit stellt dieses Konzept für die technische Realisierung eine bedeutende Rolle.

In Abschnitt 4.2 wird darauf eingegangen, wie Benutzerverhalten bewertet werden kann. Hierbei liegt der Fokus auf durch die ELAI oder andere Adaptivitätssysteme vorgenommenen Bewertungen von Benutzerverhalten, um adaptive Eingriffe in Serious Games zu unternehmen. Da die später vorgestellte Implementierung in der Lage sein soll, derartige adaptive Eingriffe durch die ELAI systematisch zu überprüfen und verifizieren, ist dies auch für die Implementierung von Interesse.

In den darauf folgenden Kapiteln wird auf weiterführende Anwendungsmöglichkeiten eingegangen, die sich durch die Standardisierung von Benutzerverhalten ergeben. Abschnitt 4.3 beschreibt, wie Verhaltensmuster aus Nutzerverhalten extrahiert werden kann. In Abschnitt 4.4 wird darüber hinaus diskutiert, ob und wie entsprechend solcher Verhaltensmuster komplette Sequenzen von Benutzeraktionen automatisch oder per Hand erzeugt werden können. Um eine systematische Verifikation eines Adaptivitätssystems zu ermöglichen, ist es notwendig, Nutzerverhalten zu Testzwecken ohne manuelles Spielen eines Serious Games erzeugen zu können.

4.1. Formale Definitionen für Standardisierung

Im Folgenden wird ein Entwurf präsentiert, der eine Standardisierung von Nutzerverhalten in ein klar strukturiertes Format ermöglicht. Der Entwurf orientiert sich an der xAPI und soll dementsprechend serialisierbar sein.

4.1.1. Definition Nutzeraktion

Eine Nutzeraktion N, im Folgenden auch als Aktion bezeichnet, ist ein Tupel $N = (A, V, O)$ aus drei Komponenten:

- Ein Akteur A, welcher die Nutzeraktion ausführt. Beispiel: $A = \textit{John Doe}$
- Ein Verb V, welches die Art der Aktion beschreibt. Beispiel: $V = \textit{klicken}$
- Ein Objekt O, auf welchem der Akteur agiert. Beispiel: $O = \textit{Schaltfläche}$

Damit orientiert sich die Definition einer Nutzeraktion an den Statements aus xAPI. Im Folgenden wird der Begriff Nutzeraktion oder Aktion im Kontext zum Entwurf für Standardisierung verwendet, während der Begriff Statement im Kontext zu xAPI, besonders später im Hinblick auf die Implementierung, verwendet wird.

In der Spezifikation der xAPI sind neben Akteur, Verb und Objekt weitere mögliche Parameter definiert, insbesondere ein *Ergebnis*-Objekt. Dieses ist bei der Bewertung notwendig und ermöglicht Adaptivität, indem eine Aktion als konstruktiv oder unkonstruktiv für den Spielfluss bewertet wird. Auf die Möglichkeiten der Bewertung von Aktionen wird genauer in Abschnitt 4.2.1 eingegangen. Zunächst seien Aktionen vereinfachend nur durch die drei genannten Komponenten modelliert.

Außerdem seien wie folgt Mengen definiert: $\mathscr{A}$ enthält alle möglichen Akteure, $\mathscr{V}$ enthält alle möglichen Verben und $\mathscr{O}$ enthält alle möglichen Objekte. Damit ergibt sich die Menge aller möglichen Aktionen $\mathscr{N}$ mit

$$\mathscr{N} = \mathscr{A} \times \mathscr{V} \times \mathscr{O}.$$

4.1.2. Definition Szenario

Einzelne Nutzeraktionen liefern ohne Kontext eher wenig Potential für Analysen des Benutzerverhaltens zu adaptiven Zwecken. Da in der Realität immer eine Reihe von Aktionen vorliegen, welche in Abhängigkeit zueinander geschehen, wird auch dies hier modelliert. Insbesondere lassen sich interessantere Informationen aus einer Reihe von Aktionen extrahieren, als nur aus einer einzelnen Aktion, da sich erst durch den Kontext und die Abhängigkeiten der Aktionen aussagekräftige Ergebnisse ableiten lassen.

Eine Reihe von Aktionen, die zeitlich aufeinander folgen, werden im Folgenden als Szenario S bezeichnet. Ein Szenario ist dabei ein Tupel aus einer Menge von Aktionen A, und einer Zeitfunktion T, welche die Aktionen zeitlich ordnet. Die Zeitfunktion ist nicht injektiv, sodass Aktionen auch zeitgleich geschehen können.

$$\begin{aligned} S &= (A, T) \\ A &= \{a_i | i \in \mathbb{N}, a_i \in \mathscr{N}\} \\ T &: A \rightarrow \mathbb{N}_0 \end{aligned}$$

Durch die Zeitfunktion, die jeder Aktion eine natürliche Zahl als Zeitpunkt des Geschehens zuordnet, ist die Reihenfolge der Aktionen durch die Strenge Totalordnung festgelegt, die durch die natürlichen Zahlen gegeben ist.

So lässt sich für je zwei beliebige Aktionen a_1 und a_2 feststellen, welche zeitlich vorher geschieht oder ob beide zeitgleich stattfinden.
Beispiel. Für drei Aktionen $a_1, a_2, a_3 \in A$ sowie einer Zeitfunktion T mit

$$\begin{aligned} T : A &\rightarrow \mathbb{N} \\ a_1 &\mapsto 0 \\ a_2 &\mapsto 2 \\ a_3 &\mapsto 2 \end{aligned}$$

gilt $T(a_1) < T(a_2)$, $T(a_1) < T(a_3)$ und $T(a_2) = T(a_3)$.

4.2. Verifikation der Bewertung von Nutzerverhalten

Das Ziel von Adaptivitätssystemen wie der ELAI liegt darin, durch adaptive Eingriffe den Spielverlauf entsprechend den Bedürfnissen des Spielers anzupassen. Dazu muss das Spielverhalten des Spielers analysiert und verarbeitet werden. Im Folgenden wird beschrieben, wie die ELAI solche Bewertungen vornimmt, wie diese Bewertungen im Spiel zwecks Adaptivität integriert werden und wie solche Bewertungen systematisch verifiziert werden können.

4.2.1. Bewertung von Aktionen und Szenarien

Um Nutzerverhalten zu bewerten, können einzelnen Aktionen ein numerischer Wert entsprechend ihrer Konstruktivität im Spielverlauf zugeordnet werden: $\mathscr{N} \mapsto [0, 1]$. In xAPI existiert neben den notwendigen Komponenten mitunter ein optionales Attribut namens *Ergebnis* (*result*). Hier kann eine Bewertung der Aktion bereits durch den ELAI-Adapter innerhalb des Spiels in Abhängigkeit des Spielzustandes erzeugt und mit der Aktion verknüpft werden. Es handelt sich dabei meist aber eher um weniger aussagekräftige Wertungsdaten, die nur im Kontext zum aktuellen Spielverlauf stehen. Im Spiel SaFIRa gibt dieser Wert beispielsweise an, wie weit der Spieler noch vom Ziel entfernt ist [Bie16].

Sobald die ELAI die Aktionsdaten einschließlich der zugehörigen Ergebnis-Felder empfängt, verarbeitet diese die Daten, um global gültige Wertungsdaten herauszuarbeiten und Empfehlungen für Adaptionsmaßnahmen zu bestimmen. Diese durch die ELAI berechneten Daten werden als *Performance-Daten* (oder *Performance-Score*) bezeichnet [SR17b]. Bei diesen Daten handelt es sich um verschiedene Parameter, die direkt in der Adaption berücksichtigt werden können. Ein solcher Parameter ist zum Beispiel „skillLevel“, welcher bei SaFIRa nach jeder Spielinstanz neu berechnet und verwendet wird, um zu Spielbeginn den Schwierigkeitsgrad zu spezifizieren. [Lei16]

Für eine detaillierte und korrekte Analyse des Benutzerverhaltens ist hierbei weiterer *Kontext* für die Bewertung der Aktion notwendig. Die ELAI bewertet Aktionen sowohl unter Berücksichtigung des bisherigen Spielverlaufs, also den zuvor registrierten Aktionen, als auch unter Berücksichtigung des Spielerprofils, welches über verschiedene Spieldurchläufe von Serious Games zu einer Person gespeichert wird.

Die durch die ELAI berechneten Performance-Daten können damit als Abbildung der Form

$$perf : \mathcal{N} \times \text{Kontext} \rightarrow [0, 1]^* \tag{4.2.1.1}$$

aufgefasst werden.

Die Performance-Daten und Empfehlungen werden von der ELAI zurück an das Serious Game gesendet, um den Spielverlauf entsprechend zu adaptieren. Ein Beispiel für einen Verlauf von Performance-Daten aus den Aktionen eines Szenarios liefert Abbildung 4.1.

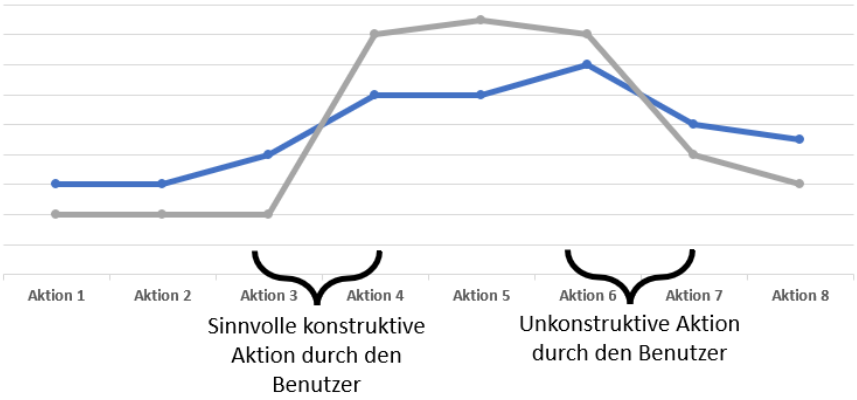


Abbildung 4.1: Verschiedene Performance-Daten können für einzelne Aktionen eines Szenarios aufgezeichnet werden.

4.2.2. Verifikation von Bewertungen

Ein integrales Ziel dieser Arbeit ist das Verifizieren der Antworten der ELAI, um deren Funktionsweise zu überprüfen. Diese Software soll getestet werden können, ohne ein daran angeschlossenes Serious Game manuell spielen zu müssen. Dabei ist vor allem wichtig, dass die ELAI deterministisch getestet werden kann. Dadurch kann sie auch im Laufe ihrer Entwicklung auf Regressionsfehler getestet werden.

Die übliche Vorgehensweise bei Komponententests (Unit-tests) ist dabei die Entwicklung von externen klar definierten Testszenarien, die in der Software durchlaufen werden. Das Verhalten der Software wird dabei mit erwarteten Reaktionen abgeglichen. Wenn die Software sich nicht wie erwartet verhält, wird sie als fehlerhaft empfunden und der Test gilt als fehlgeschlagen. Wenn die Software wie erwartet funktioniert, ist der Test erfolgreich. Das dabei erwartete Verhalten der Software wird im Vorhinein definiert. So kann zum einen während der Entwicklung der Software diese auf Vollständigkeit und Korrektheit von neu implementierten Features getestet werden, außerdem kann zuvor bestehende Funktionalität nach Abänderungen der Software darauf überprüft werden, ob sie noch wie zuvor funktioniert. Bei letzterem handelt es sich um Regressionstests. [Ham04]

Die ELAI kann auf eine ähnliche Vorgehensweise getestet werden. Sie kann von dem Serious Game abgekoppelt werden, und ein fest definiertes Szenario kann an sie gesendet werden. Das Verhalten der ELAI in Form ihrer Antworten von Adaptivitätsparametern kann dann entsprechend erwarteten Reaktionen überprüft werden.

Dieses Vorgehen kann beliebig wiederholt werden, sodass die ELAI entsprechend funktionaler Anforderungen getestet und im Nachhinein Regressionsfehler entdeckt werden können.

Trivialerweise können hierfür fest definierte Szenarien als Eingabe verwendet werden, die sich über die Testdurchläufe hinweg nicht verändern. So kann ein deterministisches Testumfeld erreicht werden, wie es auch bei Komponententests gegeben ist.

Im vorigen Abschnitt ist in Formel 4.2.1.1 eine Abstraktion für Performance-Werte als Abbildung *perf* beschrieben. Zu Verifikationszwecken kann eine Abbildung *perf'* derselben Form verwendet werden, die statisch festgelegte Sollwerte definiert, mit denen dann die Ergebnisse der tatsächlichen Performance-Daten aus *perf* abgeglichen werden.

Diese Performance-Daten gelten dann entsprechend der Verifikation für ein Szenario $S \in \mathscr{S}$ und ein beliebig kleines $\varepsilon \in \mathbb{N}_+$ genau dann als korrekt, wenn Folgendes gilt:

$$\exists \text{Kontext} : \forall N \in S : |perf'(N, \text{Kontext}) - perf(N, \text{Kontext})| \leq \varepsilon \tag{4.2.2.1}$$

Es wird also überprüft, ob die Differenz zwischen allen Performance- und Sollwerten kleiner als ein Toleranzwert ε ist. So kann selbst ein Satz von Performane-Daten als korrekt bewertet werden, der nicht komplett identische Werte hat. Wählt man $\varepsilon = 0$, so werden nur komplette Übereinstimmungen der Datensätze als korrekt gewertet.

Diese Formel wird auch in der später vorgestellten Implementierung zur Verifikation der ELAI verwendet. Die Abbildung *perf'* beschreibt hierbei die in Abschnitt 5.4 erwähnten Referenzwerte.

Abbildung 4.2 illustriert wie die Verifikationsmechanik die Abweichung der durch die ELAI generierten Werte von den durch die Tests erwarteten Sollwerte vergleicht.

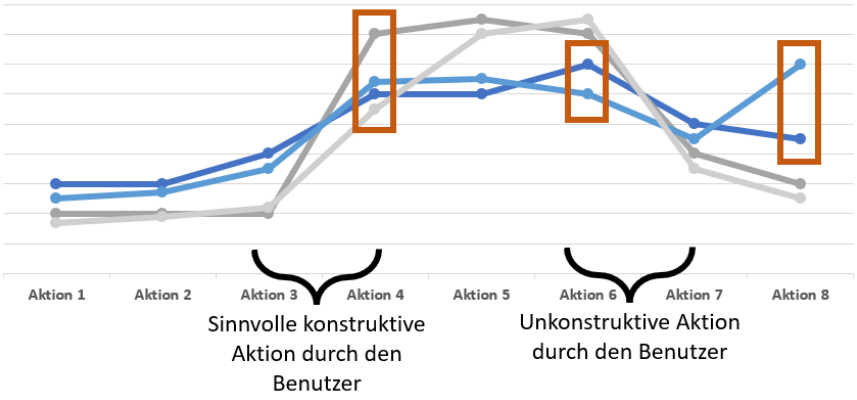


Abbildung 4.2: Die Verifikationsmechanik nimmt die Adaptionswerte der ELAI auf, vergleicht sie mit durch die Tests definierten Sollwerte und visualisiert beide auf einem Graphen. Zu große Abweichungen der aufgenommenen Werte von den Sollwerte werden als fehlgeschlagene Tests interpretiert.

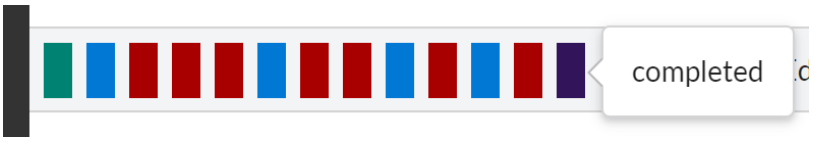
4.3. Wie können Muster in Nutzerverhalten erkannt werden?

Da es sich bei einem Szenario, also einem sequentiellen Ablauf von Benutzeraktionen, um eine komplexe Datenstruktur handelt, ist es möglich, Muster im Verhalten herauszuarbeiten. Das kann auf verschiedene Weisen Anwendung finden: Aus extrahierten Mustern können detaillierte Analysen erstellt werden, die nicht zuletzt auch zu Zwecken der Adaptivität genutzt werden können. Wenn ein konkretes Muster von Aktionen sich häufig wiederholt, kann dies als Indiz dafür aufgefasst werden, dass der Benutzer sich seiner Aufgabenstellung in der Anwendung nicht mehr bewusst ist. Im Sinne der Verifikation von Adaptivitätslogik können solche Muster-basierten Analysen außerdem verwendet werden, um Nutzerverhalten automatisiert zu generieren. Auf Letzteres wird in Abschnitt 4.4.2 genauer eingegangen.

4.3.1. Manuelles Erkennen von Mustern

Ein Szenario, welches den Spielverlauf eines Serious Games beschreibt, besteht häufig aus überschaubar vielen Aktionen. Wenn man bereits über aufgezeichnete Benutzerdaten verfügt, ist es manuell möglich, solche Muster zu erkennen. Die in Kapitel 5 beschriebene Implementierung zeigt zu einem Szenario eine Kurzübersicht über alle darin enthaltenen Aktionen. Diese ist in Abbildung 4.3 illustriert.

(a) Wiederkehrende Muster von Aktionsgruppierungen lassen sich leicht erkennen und sind hier mit einem schwarzen Rahmen dargestellt.



(b) Die Implementierung dieser Arbeit visualisiert auf ähnliche Weise jede Aktion eines Szenarios.

Abbildung 4.3: Visuelle Darstellung einer Sequenz von Aktionen.

4.3.2. Automatisches Mustererkennen nach Rashidi u.a.

Muster von Aktionen in Szenarien lassen sich auch auf automatisierte Weise extrahieren. Dazu können verschiedene Pattern-Mining Algorithmen angewandt werden. In der Arbeit von Rashidi u.a. ist eine Methode zum Finden von Aktivitätsmustern im Verhalten

von Menschen basierend auf Sensordaten beschrieben [RC10]. Dort werden im Kontrast zu dieser Arbeit keine komplexen Aktionsdaten verarbeitet, sondern leichtgewichtige Sensordaten der Form (s, t), jeweils mit einem Sensor s und einem Zeitstempel t.

Um Muster zu extrahieren, werden Sequenzen von solchen Datentupeln mithilfe eines Pattern Growth Algorithmus verarbeitet. Es werden *Allgemeine Muster* (engl. *general patterns*) gebildet, die die verschiedene Variationen von Mustern beinhalten. Die zu einem Allgemeinen Muster gehörenden Variationen haben eine zueinander ähnliche Struktur von Sensordaten, aber verschiedene Aktionsreihenfolgen.

Dann wird das Prinzip der *Minimum Description Length* (MDL, auf Deutsch *minimale Beschreibungslänge*) verwendet, demnach eine Struktur einer höheren Ordnung folgt, wenn sie über eine minimale Parametrisierung verfügt [Ris78]. Zu den Allgemeinen Mustern und den zugehörigen Variationen werden dann Kompressionswerte berechnet, um die Relevanz der Muster zu bestimmen. Der Wert der Kompression eines Allgemeinen Musters a über einen Zeitrahmen T ist durch Formel 4.3.2.1 beschrieben, während Formel 4.3.2.2 den Kompressionswert einer Variation a_i über einen Zeitrahmen T definiert. Dabei stellt B_T[1] einen Datensatz von Sensordaten im Zeitrahmen T dar, L bezieht sich auf die Beschreibungslänge entsprechend der Minimum Description Length und Γ bezieht sich auf die Kontinuität der Sensorwerte innerhalb des Musters.

$$\alpha_T(a) = \frac{L(B_T) \cdot \Gamma_a}{L(a) + L(B_T|a)} \tag{4.3.2.1}$$

$$\beta_T(a_i) = \frac{(L(B_T|a) + L(a)) \cdot \Gamma_{a_i}}{L(B_T|a_i) + L(a_i)} \tag{4.3.2.2}$$

Ein Allgemeines Muster a wird über einer Zeitspanne T dann als *interessant* bewertet, wenn der zugehörige Kompressionswert $\alpha_T(a)$ größer als ein minimaler Kompressionswert σ_g ist. Des Weiteren gilt eine Variation a_i über einer Zeitspanne T als interessant, wenn der zugehörige Kompressionswert $\beta_T(a_i)$ größer als der durchschnittliche Kompressionswert aller Variationen eines Allgemeinen Musters a über einer Zeitspanne T, wie in Gleichung 4.3.2.3 formuliert, ist.

$$\tilde{\beta}_T(a) = \frac{1}{n_a} \cdot \sum_{i=1}^{n_a} \beta_T(a_i) \tag{4.3.2.3}$$

4.3.3. Anwendung auf komplexe Verhaltensszenarien

Die durch Rashidi u.a. vorgestellte Methode findet Muster in Sequenzen von Benutzeraktivitäten [RC10]. Grundlegend lässt sich dieses damit auch auf das hier präsentierte Konzept von Szenarien und komplexen Aktionen übertragen. Der größte Unterschied liegt in der Modellierung der Aktionen: Während hier eine Aktion eine komplexe Datenstruktur mit den Parametern Akteur, Verb und Objekt ist, enthält ein Sensor-Datentupel nach Rashidi u.a. nur die Bezeichnung des Sensors sowie einen Zeitstempel. Durch die für ein

[1] Der Datensatz B_T wurde vor der Anwendung des Pattern Growth außerdem um Sensordaten von selten feuernden Sensoren reduziert. [RC10]

Szenario definierte Zeitabbildung T sind sowohl Aktionen aus dem hier vorgestellten Szenario-Modell als auch Sensordaten nach Rashidi u.a. zeitlich geordnet.

Dennoch lässt sich die Idee des Verfahrens auf das Szenario-Modell anwenden. Wichtig ist dabei, dass für eine (Teil-)Sequenz von Aktionen, also ein Muster, eine minimale Beschreibungslänge entsprechend der Minimum Description Length bestimmt werden kann. Dann kann Pattern-Mining auch auf größeren Mengen von aufgezeichneten Szenarien betrieben werden, um prominente Muster im Nutzerverhalten zu extrahieren.

4.4. Wie kann Nutzerverhalten simuliert werden?

In Abschnitt 4.2 ist beschrieben, wie man die Funktionalität eines Adaptivitätssystems wie der ELAI mit einem Verfahren verifizieren kann, welches Komponenten- bzw. Unit-Tests ähnelt, wie sie aus der Softwareentwicklung bekannt sind. Hierzu wird simuliertes Nutzerverhalten als Eingabe in das Adaptivitätssystem übergeben und die zurückgegebenen Performance-Werte werden auf erwartete Eigenschaften überprüft. In diesem Abschnitt wird auf das künstliche Erzeugen des Nutzerverhalten eingegangen, das hierbei als Input verwendet werden soll.

4.4.1. Manuelles Erzeugen von Szenarien

Um auf vergleichsweise unkomplizierte Weise ein Szenario von Benutzeraktionen zu erzeugen, kann dieses manuell über einen Editor erstellt werden. Die später vorgestellte Implementierung enthält einen grafischen Editor, der in Abschnitt 5.2 beschrieben ist. Dieser ermöglicht das manuelle Bearbeiten kompletter Szenarien. So können zur Verifikation der ELAI verschiedene unabhängige Szenarien erstellt werden, die konkrete Funktionalitäten dieser abprüfen. Abbildung 4.4 zeigt einen Ausschnitt aus der entstandenen Implementierung ElaiSim, in dem aus einem Szenario die einzelnen Aktionen bearbeitet werden können.

Auf diese Weise können klar strukturierte Szenarien definiert werden, die eine deterministische Testumgebung für ein zu testendes Adaptivitätssystem liefern.

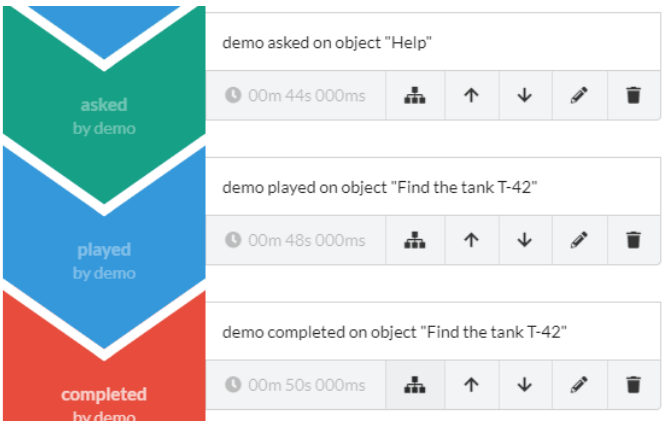


Abbildung 4.4: Sequenz von Aktionen im grafischen Editor von ElaiSim

4.4.2. Ausblick: Automatische Generation von Szenarien

Über die zuvor beschriebenen deterministischen Testfälle hinaus ist außerdem interessant, wie Szenarien automatisch generiert werden können um das Adaptivitätssystem auf dynamische Weise zu testen.

In Abschnitt 5.5 ist beschrieben, wie in der Implementierung von ElaiSim ein Szenario dynamisch abgewandelt werden kann. Dazu können zum Beispiel einzelne Werte wie etwa das *Ergebnis*-Feld einer Aktion zufällig neu berechnet werden. Während die Aktionen zwar dynamisch beliebig modifiziert werden können, ist das künstliche Erzeugen von kompletten Szenarien dadurch nicht möglich.

Um zufällige Szenarien zu erzeugen, können häufig auftretende Muster von Benutzeraktionen verknüpft werden, sodass sinnvolle komplexe Abläufe von Aktionen entstehen. Abschnitt 4.3 beschreibt, wie Pattern-Mining Methoden verwendet werden können, um häufige Muster von Benutzeraktionen aus Szenarien zu extrahieren. Für die Musterextraktion ist ein möglichst großer Datenbestand an bestehenden Szenarien notwendig. Diese Szenarien können aus tatsächlichen Spieldurchläufen des Serious Games durch verschiedene Personen aufgenommen werden, um einen zuverlässigen Datenbestand als Grundlage für das Pattern-Mining zu erhalten. Aus diesen können dann Muster extrahiert werden, welche wiederum zur automatischen Erzeugung von neuen Szenarien benutzt werden können. Hierdurch ergeben sich neue zufällige Szenarien, die in zukünftigen Testdurchläufen das Adaptivitätssystem auf dynamische Weise testen, ohne dass die Testfälle dazu manuell definiert werden müssen.

Bei der Verifikation ist entsprechend auch eine dynamischere Vorgehensweise denkbar, die zufälligen Szenarien besser gerecht wird. Statt wie in Formel 4.2.2.1 auf hart festgelegte Sollwerte zu prüfen, macht eine dynamische Überprüfung der Werte in Abhängigkeit des erzeugten Szenarios Sinn. Genauso wie aus Szenarien Muster von Verhaltensweisen extrahiert werden können um neue Szenarien zu generieren, könnte man entsprechend aus den Performance-Daten der ELAI Muster extrahieren, auf die bei zukünftigen Durchläufen geprüft wird.

Eine derartige Logik existiert in der Implementierung von ElaiSim nicht, ist aber durchaus als weiterführende Entwicklung denkbar. Am Wichtigsten ist dazu vor allem die Pattern-Mining Logik, die auch als separate Anwendung realisiert werden kann. Die extrahierten Muster könnten dann von ElaiSim importiert und zur Generierung von Szenarien verarbeitet werden.

KAPITEL 5

Implementierung eines Simulators für Nutzerverhalten

5.1. Konzept und Architektur

Das praktisch zu realisierende Ziel dieser Arbeit ist eine Lösung, welche die realistische Erzeugung und Editierung von Benutzerverhalten ermöglicht. Gleichzeitig sollen bestehende Anwendungen wie die ELAI und verschiedene Serious Games mit der Lösung verknüpfbar sein, um die ELAI systematisch verifizieren zu können. Dazu wurde ein Simulator mit der Bezeichnung *ElaiSim* (kurz für *ELAI Simulator*) implementiert, der an diese Programme angekoppelt werden und sie verifizieren kann.

Dabei wurden folgende funktionale und nicht-funktionale Anforderungen spezifiziert:

- Szenarien von Nutzerverhalten können in einem grafischen Editor visualisiert und bearbeitet werden.
- Szenarien können aus bestehenden Serious Games wie *SaFIRa* und *Lost Earth 2307* aufgenommen werden.
- Szenarien werden in einem standardisierten Format gespeichert.
- Szenarien können als simulierte Eingabe an die ELAI gesendet werden.
- Die Adaptivitätsantworten der ELAI werden durch ElaiSim visualisiert und maschinell verifiziert.
- ElaiSim ist als eigenständige Software ohne Installationsroutinen nutzbar.

Die entwickelte Lösung ist mit der Programmiersprache *TypeScript* entwickelt und verwendet *React* zur Darstellung der grafischen Oberfläche und *Redux* zur internen Zustandshaltung. Die verschiedenen Technologien sind in Abschnitt 3.5 beschrieben. Zur standardisierten Speicherung der Verhaltensmuster wird das in Kapitel 3.3 erläuterte Format *Experience API* (xAPI) genutzt, das ebenfalls von der ELAI verwendet wird.

Um eine eigenständig lauffähige Version der Software zu erhalten, wurde das in Abschnitt 3.5.6 beschriebene Framework *Electron* verwendet. Dadurch wurde ein ausführbares Programm erzeugt, welches alle Abhängigkeiten einschließlich *Node.js* und *Chromium* (dem Browserkern von *Google Chrome* zur Darstellung der grafischen Oberfläche) enthält.

ElaiSim besteht aus getrenntem Frontend und Backend. Dadurch kann die Anwendung auch in gewöhnlichen Webbrowsern geöffnet werden. Das Frontend ist neben der grafischen Oberfläche auch für die Mehrheit der Anwendungslogik zuständig. Das Backend regelt nur funktionale Anforderungen, denen das Frontend als Webapplikation im Browser aufgrund des begrenzten Funktionsumfangs von Webanwendungen nicht gerecht werden kann, wie Zugriffe auf das Dateisystem oder die Kommunikation mit externen Servern wie dem ELAI-Server.

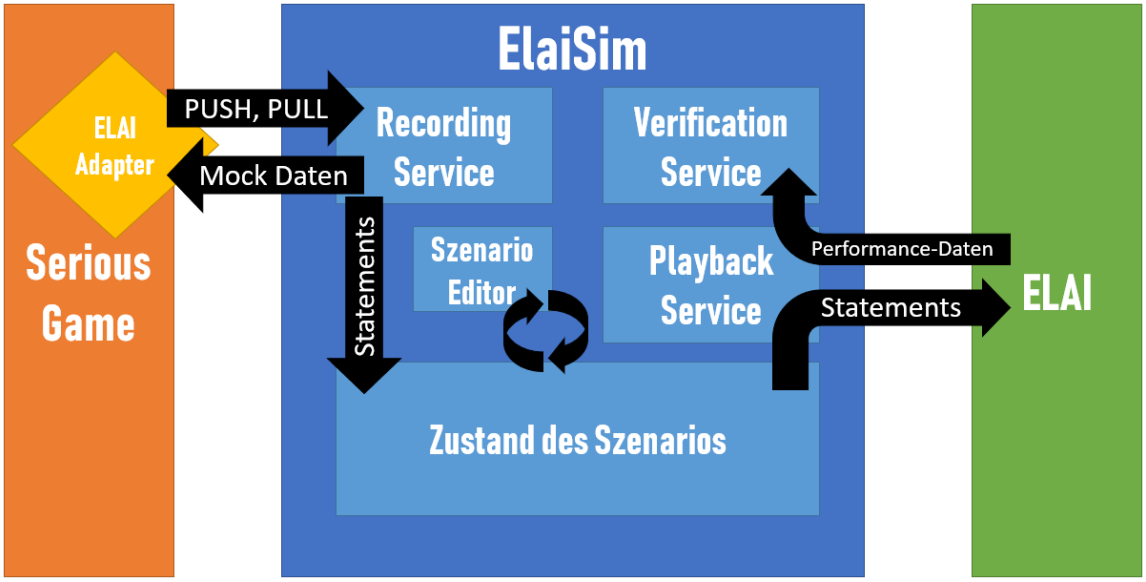


Abbildung 5.1: Die Architektur von ElaiSim, in Kombination mit angekoppeltem Serious Game und ELAI.

Die Implementierung ist außerdem in verschiedene Komponenten getrennt, die für unterschiedliche Aufgaben zuständig sind:

Szenario-Editor. Hierbei handelt es sich um den grafischen Editor, in dem Abfolgen von Nutzeraktionen bearbeitet werden können.

Recording-Service. Dieser ist dafür zuständig, Nutzerverhalten aus bestehenden Serious Games aufzunehmen und diese in dem grafischen Editor einzubetten.

Playback-Service. Diese Komponente ist für die Kommunikation mit Adaptivitätssoftware wie der ELAI zuständig und sendet die einzelnen xAPI-Statements an diese.

Postprocessing-Service. Dieser kann zusätzlich verwendet werden, um xAPI-Statements vor dem Senden durch den Playback-Service entsprechend festgelegter Regeln zu manipulieren. Dabei können einzelne Werte oder die gesamte Struktur der Statements verändert werden. So wird Interoperabilität gefördert und Randomisierung der Werte ermöglicht.

Verification-Service. Hier werden die Ergebnisdaten der ELAI empfangen und verarbeitet, nachdem diese Statements durch den Playback-Service erhalten hat. Dieser Service ist für die Verifikation der ELAI zuständig.

Die Komponenten sind als Bestandteile des Frontends integriert. Der Recording- und der Playback-Service sind zusätzlich im Backend durch REST Endpunkte vertreten, welche die Kommunikation nach außen handhaben. Die Architektur und grundlegende Funktionsweise von ElaiSim ist durch Abbildung 5.1 skizziert.

5.2. Anwendungsoberfläche und Szenario-Editor

Der Kern der Anwendung ist der Szenario-Editor, in dem ein Szenario als Abfolge von xAPI-Statements visualisiert und bearbeitet werden kann. Dieses liefert außerdem die Schnittstelle zu den anderen Funktionalitäten, die hierüber abgerufen werden können. Die Oberfläche ist in Abbildung 5.2 dargestellt. Diese besteht aus den dort eingerahmten visuellen Bausteinen:

Abbildung 5.2: Bildschirmfoto des Szenarioeditors von ElaiSim

Befehlsleiste. Hier werden einzelne Operationen, die der Benutzer durchführen kann, aufgelistet. Dazu gehört das Bearbeiten des geöffneten Szenarios, aber auch das Konfigurieren und Starten des Rekorders und des Playbacks, was in den Abschnitten 5.3 und 5.4 genauer erklärt wird. Die Befehlsleiste folgt dem in Abschnitt 5.7.2 beschriebenen Befehl-Entwurfsmuster.

Dateiliste. Über diese kann der Benutzer Szenario-Dateien direkt auf der Festplatte speichern oder von dort laden. Die Liste ist als Baumstruktur realisiert, die in ElaiSim

traversiert werden kann. Sie reflektiert den Inhalt eines Ordners auf der Festplatte, welcher frei konfigurierbar ist. Da das Frontend in einer Browser-Umgebung nicht direkt auf die Festplatte zugreifen kann, werden Zugriffe über das Backend der Anwendung geleitet. Dieses definiert als RESTful Webservice einige Endpunkte, die Festplattenoperationen ermöglichen. Ein Bildschirmausschnitt aus der Dateiliste ist in Abbildung 5.3 dargestellt.

Szenario. Im Hauptteil der Oberfläche wird der lineare Ablauf von xAPI Statements visualisiert, aus denen das aktuell geöffnete Szenario besteht. Zu jedem Statement werden Details zu dem zugehörigen Akteur und Verb sowie Knöpfe zum Bearbeiten der einzelnen Statements angezeigt. Wenn man das Fenster zum Bearbeiten eines Statements öffnet, kann man dieses entweder in einem Formular bearbeiten oder in die Ansicht eines Texteditors wechseln, in dem man direkt den JSON-Code des xAPI Statements bearbeiten kann. Hierzu wird der von Microsoft entwickelte *Monaco Editor*[1] verwendet, der das Statement in Echtzeit auf das durch RusticiSoftware definierte JSON-Schema eines xAPI-Statements[2] überprüft.

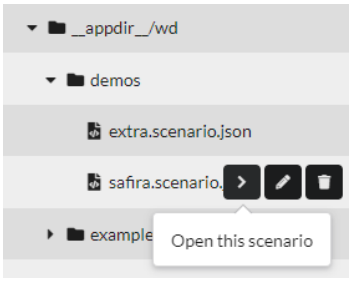


Abbildung 5.3: Dateiliste in ElaiSim

5.3. Recording-Service

In vielen Fällen ist es sinnvoll, ein Szenario nicht komplett von Hand zu erzeugen, sondern von einer externen Software aufzunehmen. Da in der gewöhnlichen Kommunikation zwischen Serious Games und der ELAI das Spiel sowieso komplette xAPI Statements als Beschreibung einer Nutzeraktion an die ELAI sendet, ist es möglich, ElaiSim als Pseudoinstanz von ELAI dazwischenzuschalten und so die vom Serious Game gesendeten Statements direkt aufzunehmen. Diese Programmlogik wird durch den *Recording-Service* (Aufnahme-Service) übernommen. ElaiSim versucht dabei, das Verhalten der ELAI nachzuahmen, sodass das Serious Game nicht mitbekommt, dass die Statements an ElaiSim gesendet werden.

Die Serious Games senden Anfragen zwei verschiedener Arten an die ELAI: Push- und Pull-Nachrichten. Push-Nachrichten enthalten eine Nutzeraktion als xAPI-Statement und werden beim Auftreten der Aktion an die ELAI gesendet, um sie über die Aktion zu

[1] `https://microsoft.github.io/monaco-editor/`
[2] `https://github.com/RusticiSoftware/TinCanSchema`

informieren. Pull-Nachrichten werden gesendet, um aktuelle Performance-Daten abzurufen. [Bie16]

Wenn ElaiSim Push-Nachrichten erhält, werden diese einfach akzeptiert und in das bearbeitete Szenario eingefügt. Dadurch wird das Verhalten der ELAI bereits reflektiert, da diese keine besondere Reaktion auf Push-Anfragen zeigt. Auf Pull-Anfragen werden leere Pseudodatensätze gesendet. Das setzt voraus, dass das Serious Game damit zurecht kommen kann und nicht von den Performance-Daten abhängig ist. Abbildung 5.4 visualisiert diesen Prozess.

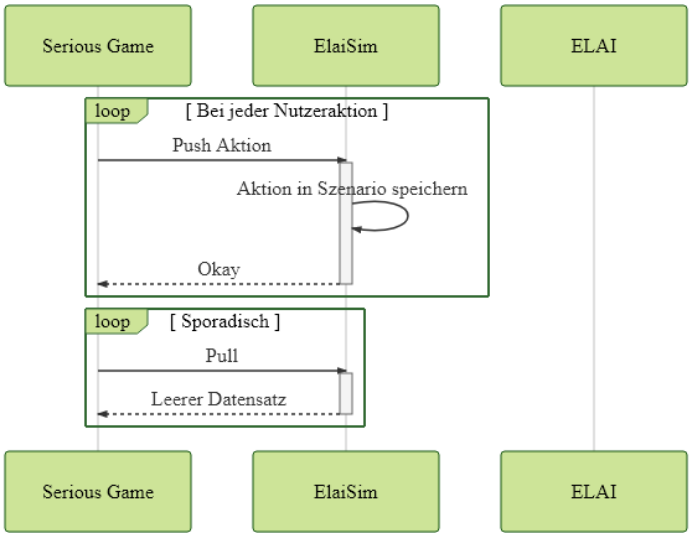


Abbildung 5.4: Die Kommunikation zwischen dem Serious Game und ElaiSim, wenn diese Nutzerverhalten aus dem Spiel aufnimmt.

Um die Schnittstelle der ELAI nachzustellen und die Anfragen abzufangen, erzeugt ElaiSim einen Webserver als RESTful Webservice, welcher dieselben REST-Schnittstellen wie die ELAI zur Verfügung stellt. Der Webserver wird durch das NPM-Paket *express*[3] realisiert. Die REST-Schnittstellen sind, um möglichst verschiedene Serious Games zu unterstützen, in der Oberfläche der Anwendung frei konfigurierbar. Auf die Weise, wie mit ElaiSim verschiedene Serious Games unterstützt werden können, wird in Abschnitt 5.6 zur Thematik Interoperabilität eingegangen. Ein Überblick über die Konfigurierbarkeit des Recording-Services liefert Tabelle A.1 im Anhang.

5.4. Playback-Service und Verification-Service

Die Grundlage zur Funktionalität der maschinellen Verifikation der ELAI stellt der *Playback-Service* (Wiedergabe-Dienst) dar. Ähnlich dazu, wie der Recording-Service an ein Serious Game gekoppelt werden kann um xAPI Datenströme aufzunehmen, kann der Playback-Service an die ELAI gekoppelt werden um xAPI Datenströme zu senden. Dazu kann das in ElaiSim erzeugte und bearbeitete Szenario verwendet werden, um die darin enthaltenen

[3] `https://npmjs.com/package/express`

xAPI Statements zu senden. Die Statements werden als *Push*-Nachricht entsprechend der in Kapitel 3.2 erläuterten Vorgehensweise gesendet. Abbildung 5.5 illustriert diesen Vorgang.

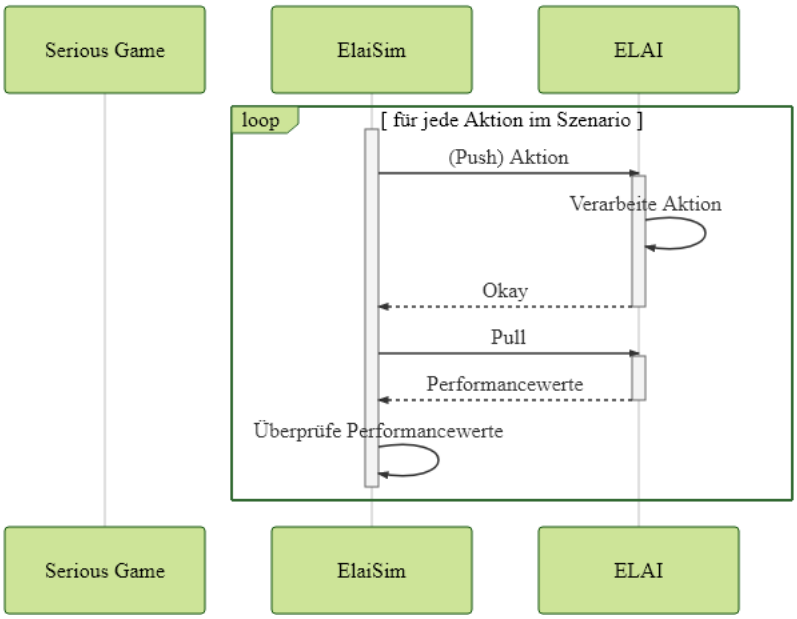


Abbildung 5.5: Die Kommunikation zwischen ElaiSim und der ELAI während der laufenden Wiedergabe von Statements. Die ELAI reagiert auf die Anfragen in der Annahme, dass es sich bei ElaiSim um ein Serious Game handelt.

In dem Szenarioeditor können konkrete Zeiten für jedes Statement spezifiziert werden. Diese werden auch bei der Aufnahme von Nutzerverhalten durch den Recording-Service entsprechend dem Zeitpunkt gesetzt, zu dem die jeweiligen Statements aufgenommen werden. Bei der Wiedergabe startet ein Zeitgeber, welcher zu den gesetzten Zeitpunkten die jeweiligen Statements sendet. Während der Wiedergabe kann diese auch pausiert und fortgesetzt werden, Statements können schrittweise durchlaufen werden oder das gesamte Szenario kann im Schnelldurchlauf gesendet werden.

Die ELAI empfängt die Nachrichten in der Annahme, dass sie von einem Serious Game gesendet wurden. Nach jedem Statement sendet ElaiSim eine zusätzliche *Pull*-Nachricht an ELAI, um Performance-Werte abzufragen. Diese werden dann durch den *Verification-Service* verarbeitet. Wenn mit dem Szenario zuvor Referenz Performance-Daten als Sollantworten der ELAI verknüpft worden sind, gleicht ElaiSim diese Solldaten mit den empfangenen Daten ab und verifiziert so die Antworten der ELAI. Falls alle Daten übereinstimmen, gilt der Durchlauf als erfolgreich. Falls dies nicht der Fall ist, berichtet ElaiSim die fehlgeschlagenen Vergleiche. Falls zuvor keine Referenz Performance-Daten verknüpft worden sind, werden die empfangenen Performance-Daten ohne Verifikation angezeigt. Der Benutzer hat dabei die Möglichkeit, die empfangenen Daten als Referenzdaten für zukünftige Durchläufe zu speichern. Diese können außerdem im Szenarioeditor manuell optimiert werden.

5.5. Postprocessing-Service

Mit der Programmlogik des Playback-Services, der in Abschnitt 5.4 erläutert wird, kann zusätzlich der Postprocessing-Service verknüpft werden, um Statements vor dem Senden dynamisch anzupassen. Die vorzunehmenden Modifikationen werden entsprechend festgelegten Regeln vorgenommen. Ein Beispiel für eine derartige Regel liefert Abbildung 5.6. Die dort dargestellte Regel beschreibt eine Abwandlung des durch xAPI definierten Ergebnis-Wertes um ein zufälliges Epsilon.

```
$[?(@.id=="c718..")].result.score.scaled: randomEpsilonOnOldValueFloat .2
```

Abbildung 5.6: Beispiel für eine Regel des Postprocessing-Services. Die Regel definiert eine Variierung des `statement.result.score.scaled` Wertes des Statements mit der ID `c718..` um $\varepsilon = 0,2$.

5.5.1. Bestandteile der Postprocessing Regeln

Jede Regel, die durch den Postprocessing-Service verarbeitet wird, um Statements anzupassen, besteht aus zwei Teilen.

Der erste Bestandteil ist eine Beschreibung des zu bearbeitenden Statements sowie des zu bearbeitenden Parameters des Statements. Im Beispiel lautet dieser Teil `$[?(@.id=="c718..")].result.score.scaled`. Hierbei wird auf die durch Stefan Goessner entwickelte Definition *JSONPath* zurückgegriffen [Goe07]. Hierbei handelt es sich um eine Abfragesprache, mit der konkrete Teilmengen einer komplexen JSON-Struktur abgerufen werden können. Die Sprache ist inspiriert durch den vom W3C definierten Standard *XML Path Language* (XPath), welcher dasselbe Ziel für *XML*-Strukturen (*Extensible Markup Language*) verfolgt [RDS17]. Mit derartigen Abfragen können nicht nur konkrete Statements per ID identifiziert werden, sondern aufgrund beliebiger Merkmale wie Verb oder Akteur. Außerdem können auch sämtliche Statements in einem Szenario mit der JSONPath Abfrage `$[*]` erreicht werden.

Zweiter Bestandteil ist ein parametrisierter Modifikator, der beschreibt wie die durch den JSONPath angesprochene Struktur abgewandelt werden soll. Dazu sind in ElaiSim die in Tabelle 5.1 aufgeführten Modifikatoren definiert, die zufällige Neubelegung der Werte oder auch komplexere Operationen ermöglichen.

5.5.2. Nutzen und Vorteile des Postprocessing-Services

Der Postprocessing-Service dient vor allem zwei Zwecken: Zum einen ermöglicht er eine Randomisierung sämtlicher Werte auf den Statements. So können beliebige Bestandteile der Statements entweder komplett zufällig oder in Abhängigkeit der alten Werte neu besetzt werden. Die ELAI lässt sich so darauf überprüfen, wie sie sich im Umgang mit nicht deterministischen Änderungen verhält.

Darüber hinaus treibt der Postprocessing-Service auch die Interoperabilität von ElaiSim voran. Die Regeln ermöglichen nicht nur die Neubelegung einzelner Werte, sondern auch

Modifikator	Beschreibung
`randomFloatBetween min max`	Überschreibt den alten Wert mit einer zufälligen Fließkommazahl zwischen den Grenzen `min` und `max`.
`randomFloat`	Überschreibt den alten Wert mit einer zufälligen Fließkommazahl zwischen null und eins.
`randomEpsilonOnOldValueFloat eps`	Addiert oder subtrahiert den alten Wert um eine zufällige Fließkommazahl zwischen null und `eps`.
`randomValueFrom ...values`	Nimmt beliebig viele Parameter, und überschreibt den alten Wert mit einem zufälligen gegebenen Parameter.
`exactValue value`	Überschreibt den alten Wert mit dem gegebenen Wert.
`exactFloat value`	Überschreibt den alten Wert mit der gegebenen Fließkommazahl.
`join structure`	Interpretiert den Parameter als JSON enkodierte Struktur und vereinigt die alte Struktur mit der neuen gegebenen. Duplikate Werte werden durch die angegebene Struktur überschrieben.
`eval code`	Interpretiert den Parameter `code` als JavaScript Funktion, welche maximal einen Parameter nimmt. Die Funktion wird mit dem alten Wert als Parameter ausgeführt und der Rückgabewert überschreibt den alten Wert.

Tabelle 5.1: Modifikatoren des Postprocessing-Services, um Komponenten von Statements abzuändern.

die Abwandlung komplexer Strukturen. Durch aufwendigere Regeln kann dadurch das komplette Statement dynamisch beim Senden in ein anderes Schema konvertiert werden, während im Editor der ElaiSim trotzdem weiterhin nach dem xAPI-Schema gearbeitet werden kann. So kann ElaiSim auch Anforderungen von anzukoppelnden Diensten gerecht werden, die Anfragen nicht nach der xAPI-Spezifikation, sondern einer beliebigen anderen Spezifikation erwarten.

Mit aufwendigeren Regeln ist es beispielsweise möglich, Statements in das *Activity Stream* Format zu konvertieren. ElaiSim stellt eine Regel, die dieses Ziel beispielhaft verfolgt, als Vorlage zur Verfügung. Ein Ausschnitt davon ist in Abbildung 5.7 dargestellt.

5.6. Interoperabilität der Anwendung

Um ElaiSim möglichst kompatibel im Hinblick auf angekoppelte Serious Games zu gestalten, wurde die Kommunikation so allgemein und individualisierbar gehalten wie möglich. Über Einstellungen innerhalb der Anwendungsoberfläche sowie Konfigurationsdateien kann die Kommunikation dabei genauer spezifiziert werden. Sowohl für das Verbinden mit Serious Games als auch bei der Verbindung mit der ELAI können grundlegende Parameter wie URL oder Port eingestellt werden. Tabelle A.1 im Anhang listet alle Einstellungen auf, mit denen die Kommunikation mit Serious Games über den Recording-Service oder die

Abbildung 5.7: Codeausschnitt einer Postprocessing Regel, welche Statements von ihrer ursprünglichen xAPI-Struktur in das Activity Stream Format konvertiert.

Kommunikation und Verifikation mit der ELAI über den Playback- und den Verification-Service konfiguriert werden können.

In der Entwicklung von ElaiSim wurde bevorzugt das von Biegemeier entwickelte Serious Game SaFIRa verwendet, da dieses einer übersichtlichen Implementierungsstruktur folgt und aufgrund gut sichtbarer Adaptivität bei der Entwicklung von ElaiSim unterstützend eingesetzt werden konnte [Bie16]. Dennoch soll durch das Konfigurationspotential von ElaiSim erreicht werden, dass man ein möglichst großes Spektrum an Serious Games an ElaiSim anschließen kann, ohne dafür den Programmcode anpassen zu müssen. Mit der Konfigurierbarkeit des Playback-Services soll außerdem möglich sein, ElaiSim ohne großen Aufwand an Änderungen in der Logik der ELAI anpassen zu können und möglicherweise sogar ElaiSim mit anderen Adaptivitätswerkzeugen zu verbinden.

Die Interoperabilität von ElaiSim wird außerdem durch den in Abschnitt 5.5 beschriebenen Postprocessing-Service vorangetrieben. Dadurch wird die dynamische Anpassung der zu sendenden Statements ermöglicht, ohne jedes Statement einzeln modifizieren zu müssen. So kann ElaiSim konkreten Anforderungen von speziellen externen Programmen gerecht werden, die entweder härtere Anforderungen bezüglich des Schemas haben oder sogar einen komplett anderen Standard erwarten: Der Postprocessing-Service ermöglicht die komplette Umstrukturierung der Statements beim Senden, sodass diese auch in ein anderes Format konvertiert werden können.

5.7. Verwendete Entwurfsmuster

In der technischen Umsetzung der Anwendung wurden verschiedene Entwurfsmuster verwendet. Hierdurch wird die Implementierung sowie die Architektur übersichtlicher und strukturierter. Im Folgenden wird kurz auf zwei der eingesetzten Entwurfsmuster eingegangen.

5.7.1. Model View Controller

Eines der prominentesten Entwurfsmuster ist das *Model View Controller* (MVC) Muster. Es empfiehlt die Separierung der Anwendungslogik in drei Teile: Datenmodell (Model), Benutzersicht (View) und Programmlogik (Controller). Dieses Muster ist vor allem bei der Entwicklung von grafischen Benutzeroberflächen von Bedeutung, da hier genau diese drei Teile implementiert werden müssen. Der schematische Aufbau des Musters ist in Abbildung 5.8 dargestellt. [KP88]

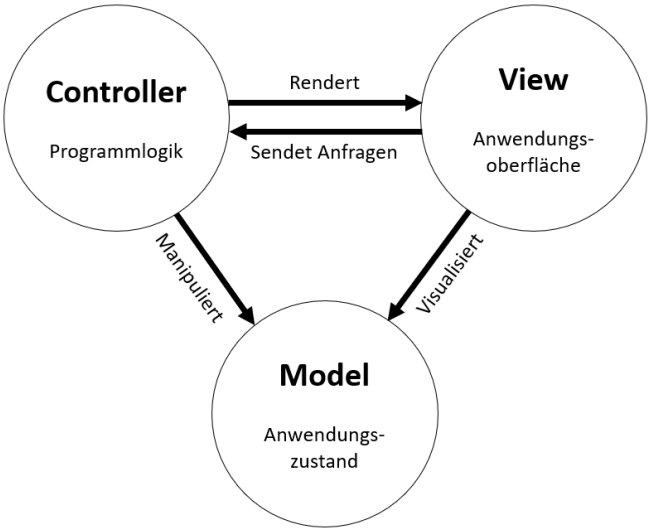


Abbildung 5.8: Entwurfsmuster „Model View Controller“ [KP88]

In der Implementierung von ElaiSim wird dieses Entwurfsmuster zu einem großen Teil bereits durch die Orchestrierung der Bibliotheken eingesetzt. ElaiSim basiert auf der JavaScript Bibliothek React, die in Abschnitt 3.5.5 erläutert wird. Diese erleichtert die Entwicklung von grafischen Oberflächen, indem sie die Kapselung von wiederverwendbaren UI-Komponenten ermöglicht. Diese UI-Komponenten können jeweils mit Anwendungslogik versehen werden, sodass hierdurch der View- und der Controller-Teil des MVC-Musters umgesetzt wird.[4]

In Verbindung zu React wurde auf die Bibliothek Redux gesetzt, mit der ein globaler Anwendungszustand sowie Zustandslogik definiert werden kann. Die Funktionsweise von Redux ist ebenfalls in Abschnitt 3.5.5 beschrieben. Mit dieser folgt die Implementierung des Anwendungszustandes von ElaiSim mittels Redux außerdem dem von Gamma u.a. definierten *Memento* Entwurfsmuster [Gam+95].

[4] Außerhalb der grafischen Benutzeroberfläche sind in ElaiSim die in Abschnitt 5.1 beschriebenen Services als eigenständige Dienste implementiert, die keine direkte Kopplung zur grafischen Oberfläche haben. Diese Services stehen damit außerhalb des Kontextes des MVC-Musters, da sie nur Programmlogik implementieren.

5.7.2. Befehl

Das *Befehl*-Entwurfsmuster ermöglicht es, eine konkrete Anfrage in einem Objekt zu kapseln [Gam+95] . Das Entwurfsmuster ist in Abbildung 5.9 visualisiert. Eine abstrakte Formulierung eines Befehls wird durch verschiedene konkrete Befehle implementiert, welche jeweils konkrete Zustandsinformationen und konkrete Programmlogik enthalten.

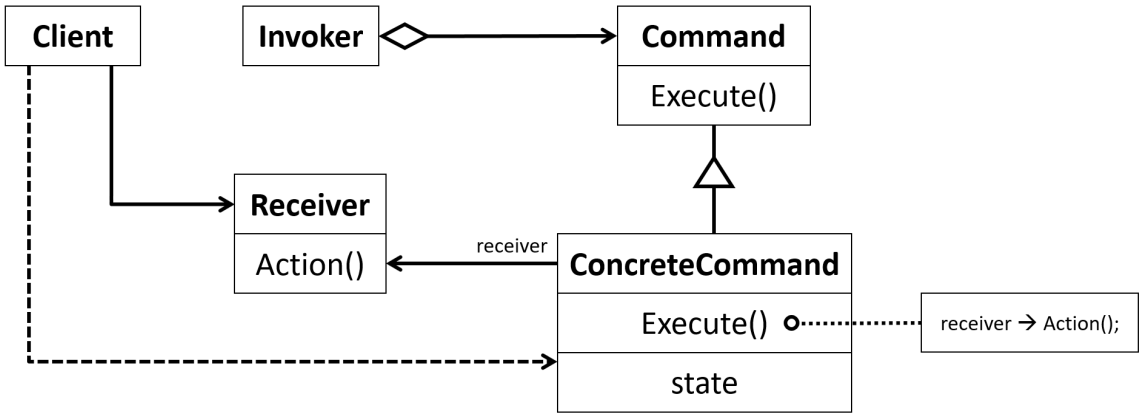


Abbildung 5.9: Entwurfsmuster „Befehl“ [Gam+95]

In ElaiSim wird dieses Muster verwendet, um eine Vielzahl von durch den Benutzer durchführbaren Operationen in Objekten zu kapseln und dynamisch in der Befehlsleiste darzustellen. Die in Abbildung 5.10 abgebildete Befehlsleiste von ElaiSim erzeugt dafür für jeden Befehl eine Schaltfläche. Die dort angezeigten Informationen wie Bezeichnung und Icon des Befehls werden aus dem Befehlsobjekt herausgezogen. Auf diese Weise sind die Befehle, die ein Benutzer durchführen kann, in einer übersichtlichen Struktur definiert und leicht erweiterbar und austauschbar. Dadurch wird außerdem die Programmlogik eines Befehls ausgelagert, während die UI-Komponente der Befehlsleiste nur noch für das Gruppieren und Anzeigen der Schaltflächen zuständig ist.

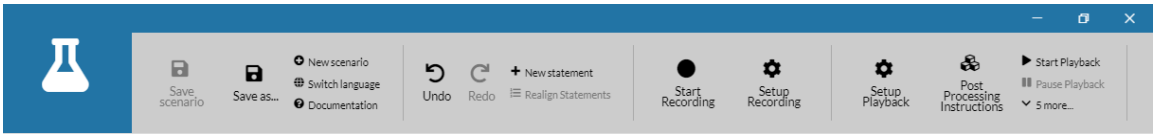


Abbildung 5.10: Befehlsleiste in ElaiSim

5.8. Fazit und Diskussion

Die Implementierung der ElaiSim ist mit der Software, die im Projektumfeld dieser Arbeit entstanden ist, kompatibel und kann über die standardisierten Schnittstellen mit diesen kommunizieren. Zu dieser Software gehören vor allem die Serious Games sowie die ELAI. Dadurch kann die Funktionalität und Korrektheit Letzterer überprüft und verifiziert werden. Durch den grafischen und textbasierten Editor sowie den Recording-Service ist

es sehr einfach, Szenarien als Testfälle zu verfassen und verwalten. Damit erfüllt die Implementierung die Zielsetzung dieser Arbeit.

Außerhalb des Projektumfeldes existieren allerdings viele Projekte, die keine standardisierten Protokolle zur Kommunikation des Benutzerverhaltens und seines Lernfortschritts verwenden. Das erschwert die Anbindung von ElaiSim an derartige Programme. Durch den Postprocessing-Service sowie die Konfigurierbarkeit ist es dennoch möglich, den spezifischen Anforderungen von Drittanbietersoftware gerecht zu werden, ohne notwendigerweise die Implementierung von ElaiSim anpassen zu müssen.

Neben den tatsächlich implementierten Features von ElaiSim sind in dieser Arbeit verschiedene Konzepte in Kapitel 4 vorgestellt und erläutert worden, die aus wissenschaftlicher Sicht für den Kontext der Arbeit zwar interessant sind, aber aufgrund der Komplexität und des Aufwands nicht in die Implementierung eingegangen sind.

Abschnitt 4.3 beschreibt, wie manuell oder automatisiert Verhaltensmuster in Szenarien von Benutzeraktionen erkannt werden können. Während ElaiSim zwar eine Kurzübersicht über alle Aktionen des Szenarios liefert, auf der unter Umständen Muster manuell erkannt werden können, liefert ElaiSim keine automatische Detektion solcher. Genauso können in ElaiSim zwar Szenarien von Benutzeraktionen manuell über den Editor erstellt werden oder aus einem Serious Game aufgenommen werden, aber nicht mithilfe von automatischen Mechanismen generiert werden.

KAPITEL 6

Anwendungsbeispiel des Simulators

Im Folgenden wird in Form eines Anwendungsbeispiels ein potentieller Ablauf beschrieben, nach dem ein Benutzer den implementierten Nutzungssimulator verwenden kann. Der beschriebene Ablauf soll die Anwendung der Applikation realistisch abbilden und ist darauf folgend mit Bildschirmfotos veranschaulicht.

6.1. Beschreibung der Anwendung

6.1.1. Starten der Anwendung

Wenn der Anwender des Simulators diesen startet, hat er zunächst die Möglichkeit, ein zuvor bearbeitetes Szenario zu öffnen oder ein Neues zu erstellen. Wenn er sich zu letzterem entscheidet, kommt er zur Ansicht „Neues Szenario erstellen“. Dort wird ihm der visuelle Editor repräsentiert, in dem die Abfolge von xAPI Statements, jeweils repräsentativ für eine Nutzeraktion, bearbeitet werden kann.

6.1.2. Aufnehmen aus externem Serious Game

Um xAPI Statements aus dem Serious Game SaFIRa aufzunehmen, öffnet der Anwender das Fenster zur Konfiguration des Recording Services, wo er den separaten Server konfiguriert, welcher später xAPI Statements des Serious Games aufnehmen soll. Indem er in der Applikation im Menü auf „Rekorder starten“ klickt, wird der Server gestartet und ein Fenster öffnet sich, welches ankommende Statements auflistet. Der Anwender startet das Spiel SaFIRa, wodurch der Recording Service die xAPI Statements abfängt. Sobald der Anwender mit der Menge der angekommenen Statements zufrieden ist, beendet er das Spiel und fügt die Statements in den Ablauf des geöffneten Szenarios ein.

6.1.3. Bearbeiten des Szenarios

Als nächstes können die einzelnen Statements für sich oder als gesamter Ablauf bearbeitet werden. Die Oberfläche erlaubt die Manipulation der Reihenfolge von den aufgenommenen Statements sowie das Löschen und Einfügen von Statements. Zu jedem Statement kann

ein Fenster geöffnet werden, in dem dessen Parameter über ein Formular oder als JSON-Rohtext bearbeitet werden können. In beiden Fällen besteht die Möglichkeit, Änderungen an sämtlichen anderen Statements mit übereinstimmenden Parametern zu reflektieren. Wenn beispielsweise in einem Statement der Name des Akteurs modifiziert wird, ändern sich, falls die Option aktiviert ist, auch bei allen anderen Statements mit demselben Akteur der Name.

Der Rohtext-Editor, in welchem der JSON-Ursprung editiert werden kann, erlaubt dem Anwender, auf einfache Weise Statements aus dem Editor heraus und hinein zu kopieren und ermöglicht eine detaillierte Bearbeitung. Fehler in der JSON-Syntax oder in der JSON-Struktur entsprechend der TinCan Spezifikation werden hervorgehoben und in der Oberfläche aufgelistet.

Über die Struktur einzelner Statements hinaus können außerdem die Statements mit Freitext kommentiert oder genau die Postprocessing-Instruktionen definiert werden, die das jeweilige Statement betreffen. In einer separaten Ansicht können auch alle Postprocessing-Instruktionen des Szenarios bearbeitet werden.

6.1.4. Abspielen des Szenarios

Sobald der Anwender das Szenario in einen Zustand gebracht hat, mit dem er zufrieden ist, kann er die Abfolge der Statements an die ELAI senden. Dazu konfiguriert er, ähnlich wie er es bei dem Recording Service gemacht hat, nun den Playback Service, welcher für das Abspielen der Statements zuständig ist. Dort definiert er die Kommunikation für PUSH und PULL Anfragen. Wenn er nun auf „Start Playback“ in der Menüleiste klickt, beginnt der Simulator, die Statements nacheinander entsprechend der Postprocessing Instruktionen abzuwandeln und als PUSH Nachrichten an die ELAI zu senden. Dazu verwendet ELaiSim die definierten Aktionszeiten, die für jedes Statement definiert sind und entweder beim Aufnehmen gemessen oder manuell im Editor gesetzt wurden. Gleichzeitigt fragt ElaiSim in vordefinierten Abständen mittels PULL Nachrichten die Performance-Werte des Benutzers von der ELAI ab.

Während das Abspielen läuft, werden die von der ELAI gelieferten Performance-Werte in einem Diagramm visualisiert. Der Benutzer kann Detailinformationen über diese abrufen, das Abspielen pausieren und einzelne xAPI Statements schrittweise abspielen. Nach dem Durchlauf können die aufgenommenen Performance-Werte mit dem Szenario verknüpft und als Referenzwerte für zukünftige Durchläufe gespeichert werden.

Zum Schluss kann der Benutzer im Editor weiterhin den Ablauf von Statements sowie die aufgenommenen Performance-Werte anpassen. Sobald sie einer Struktur folgen, mit denen er zufrieden ist, speichert der Benutzer das Szenario auf der Festplatte, von wo es später erneut im Editor geöffnet werden kann. Durch den textbasierten Speicherzustand kann das Szenario ohne Probleme mit Versionierungssoftware wie *git* versioniert werden.

6.1.5. Verifikation

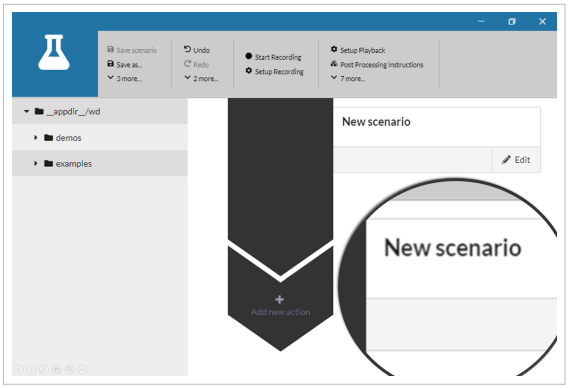
Um in einem zukünftigen Durchlauf die Funktionalität der ELAI zu überprüfen, öffnet der Benutzer das zuvor gespeicherte Szenario, konfiguriert erneut den Playback Service und startet den Ablauf. Die Statements werden wie zuvor beschrieben an die ELAI gesendet und

neue Performance-Werte werden von dieser abgefragt. Da diesmal ein Satz von Referenzdaten mit dem Szenario verknüpft ist, werden diesmal die empfangenen Performance-Werte mit den Referenzdaten abgeglichen. Entsprechend dem in Abschnitt 4.2.2 vorgestellten Konzept werden beide Datensätze verglichen. Im Ablauf werden fehlgeschlagene Vergleiche markiert und der gesamte Durchlauf wird als erfolgreich oder fehlgeschlagen bewertet, je nachdem ob währenddessen alle Datensätze übereinstimmen oder nicht.

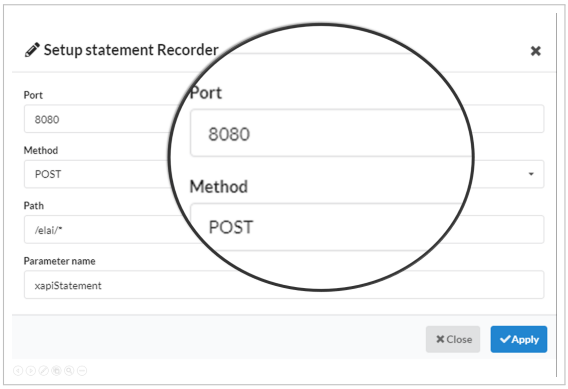
So erhält der Benutzer Informationen darüber, ob sich die ELAI entsprechend den Erwartungen verhält. Wenn beide Datensätze abweichen, weil die Logik der ELAI angepasst wurde und das neue Verhalten den neuen Erwartungen entspricht, können die Referenzdaten mit den neuen Performance Daten überschrieben werden.

6.2. Visueller Ablauf

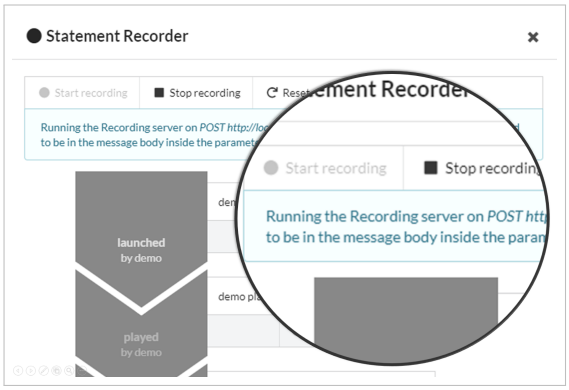
Im vorigen Abschnitt ist ein typischer Anwendungsablauf von ElaiSim skizziert. Im folgenden wird dieser Ablauf in Form einer Reihe von Bildschirmfotos visualisiert.



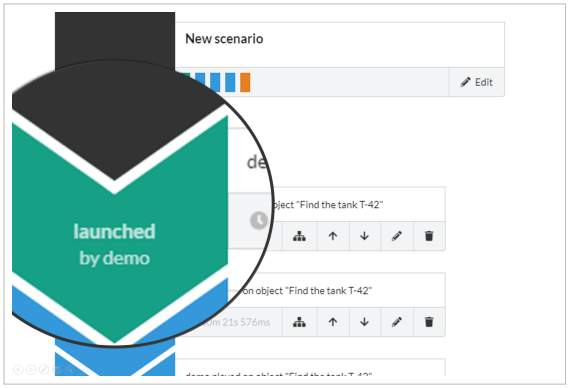
Neues Szenario erstellen
Der Benutzer öffnet ElaiSim und erstellt ein neues leeres Szenario. Eine leere Abfolge von xAPI Statements wird dargestellt.



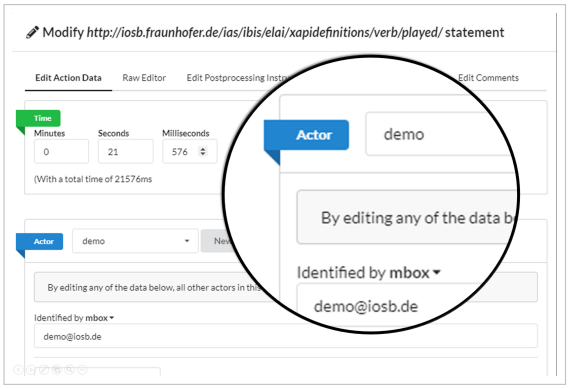
Aufnahme konfigurieren
Der Benutzer setzt die Einstellungen für die Aufnahme von Statements aus einem Serious Game. Hier wird definiert, wie das Serious Game xAPI Statements sendet.



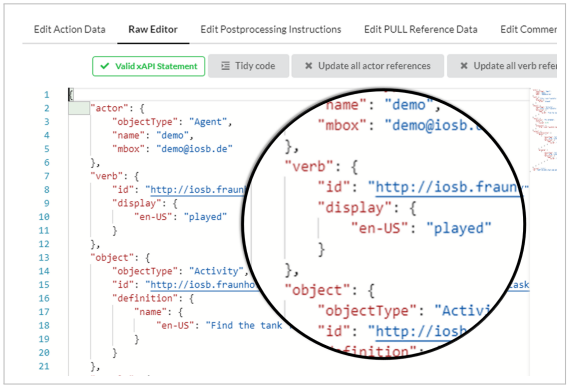
Statements aufnehmen
Die Aufnahme wird gestartet und die vom Serious Game gesendeten Statements werden in ElaiSim in einer temporären Liste dargestellt.



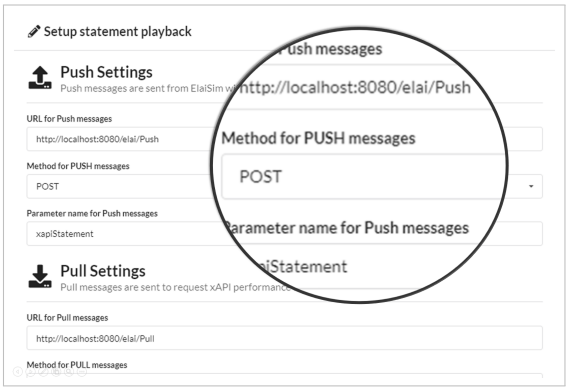
Szenario manuell anpassen
Nachdem die zuvor aufgenommenen Statements in das aktuell geöffnete Szenario eingebunden wurden, können deren Anordnung sowie jedes Statement für sich bearbeitet werden.



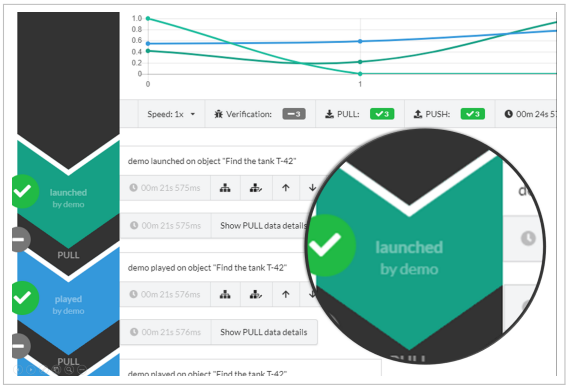
Statements bearbeiten
Einzelne Statements können in einem Formular editiert werden.



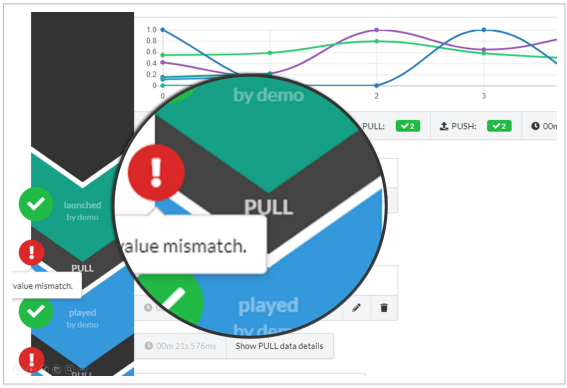
Statement Rohdaten manipulieren
Alternativ zum Formular können Statements auch in einem Rohtext Editor bearbeitet werden.



Abspielen konfigurieren
Um die Statements an die ELAI zur Verifikation zu senden, wird die Kommunikation zur ELAI spezifiziert.



Statements abspielen
Das Abspielen wird gestartet, und ElaiSim beginnt die Statements via PUSH Nachrichten zu senden und gleichzeitig mit PULL Performance-Werte abzufragen und diese zu visualisieren. Die erhaltenen Performance-Werte können dann als Referenzwerte gespeichert werden.



ELAI Rückgaben verifizieren
In zukünftigen Durchläufen werden die empfangenen Performance-Daten mit den zuvor gespeicherten Referenzwerten abgeglichen. Beide Datensätze werden visualisiert und der Durchlauf wird als erfolgreich oder fehlgeschlagen bewertet.

6.3. Diskussion

Die vorigen Abschnitte haben den typischen Anwendungsablauf von ElaiSim beschrieben. Die Anwendung ermöglicht das strukturierte Arbeiten mit Sequenzen von xAPI Statements und liefert Integrationsmöglichkeiten für sowohl Serious Games als auch Adaptivitätssystemen wie der ELAI.

Da ElaiSim das Aufnehmen und Speichern von Szenarien aus Serious Games und Performance-Daten aus dem Adaptivitätssystem ermöglicht, ist eine Verifikation von Letzterer bereits ohne manuelle Anpassung und Bearbeitung von Szenarien möglich. Trotzdem kann über den grafischen Editor die Grundlage zur Verifikation feingranular angepasst werden, um eine stabile Testumgebung zu liefern.

Als eigenständige Anwendung ohne komplexe Installationsroutinen ist sie sehr schnell in die Entwicklungsumgebung der ELAI integrierbar. So kann die Entwicklung der ELAI direkt durch automatisierte Tests unterstützt werden.

KAPITEL 7

Fazit und Ausblick

Ziel dieser Arbeit war eine konzeptionelle Standardisierung mit einer Betrachtung der daraus folgenden Möglichkeiten zur maschinellen Verifikation der ELAI. Darüber hinaus sollte eine praktische Implementierung des Nutzungssimulators *ElaiSim* realisiert werden, der dem vorgestellten Konzept folgt und eine solche Verifikation ermöglicht.

Um einem einheitlichen Standard bei der Modellierung von Nutzerverhalten zu folgen, wurden verschiedene Protokolle vorgestellt und in der Realisierung auf die xAPI zurückgegriffen, die in der bestehenden Implementierung der ELAI bereits Anwendung findet. Die notwendigen Features zur Verifikation der ELAI, einschließlich dem Aufnehmen und manuellen Bearbeiten von Nutzerverhalten, dem Einspielen von Verhalten sowie dem Überprüfen und Verifizieren der Adaptivitätsantworten der ELAI wurden alle umgesetzt.

ElaiSim wurde mit einem Fokus auf Interoperabilität entwickelt. Durch die Verwendung von Plattform-unabhängigen Technologien wie NodeJS und Electron ist die Anwendung auf verschiedenen Plattformen unabhängig von vorinstallierten Frameworks lauffähig. Vor allem soll die Anwendung aber in Kombination mit verschiedener externen Software kommunizieren und arbeiten können. Die ElaiSim muss eine Verbindung sowohl mit Serious Games zum Aufnehmen von Nutzerverhalten als auch mit Adaptivitätssoftware zum Einspielen des Verhaltens und der Verifikation der Antworten aufnehmen können. Die Kommunikation mit solcher externen Software ist dabei individuell anpassbar, um eine möglichst verschiedene externe Anwendungen zu unterstützen. Beim Abspielen von Verhalten ist das Ausgabeformat durch Anwendung des Postprocessing Services individualisierbar. Damit konnten alle zu Beginn gesetzten Ziele und Anforderungen innerhalb des vorgegebenen Zeitrahmens erfüllt werden.

In späteren Arbeiten ist eine Realisierung der Konzepte denkbar, die in dieser Arbeit erläutert, aber nicht implementiert worden sind. In Abschnitt 4.3 wurde auf ein bestehendes Verfahren zur Erkennung von Mustern in Benutzerverhalten erklärt. Eine derartige Mustererkennung ist auch in Weiterentwicklungen von ElaiSim möglich. Damit können zum einen statistische Analysen auf Daten von Benutzerverhalten geführt werden, zum anderen können erkannte Muster zur automatischen Generierung von Verhalten eingesetzt werden. Automatische Erzeugung von Benutzerverhalten kann eine dynamische und intelligente Testumgebung für ELAI und ähnliche Software ermöglichen. Abschnitt 4.4.2 geht genauer auf diese Idee ein.

Die Konzepte können als separate Anwendungen oder in der bestehenden Implementierung eingebaut werden. Durch die Kapselung der Implementierung in verschiedenen Services sowie der Anwendung von strukturierenden Frameworks wie React und Redux sind Erweiterungen der ElaiSim unkompliziert realisierbar.

Literatur

[Abt87] Clark C Abt. *Serious Games*. University Press of America, 1987.

[Adv09] Advanced Distributed Learning (ADL). "Sequencing and Navigation (SN) SCORM 2004 4th Ed. v1.1". In: (2009), S. 234. ISSN: 07366205. DOI: 10.2144/000113102.

[Adv18] Advanced Distributed Learning (ADL). *xAPI-Spec*. 2018. URL: https://github.com/adlnet/xAPI-Spec (besucht am 11. 08. 2018).

[Arn+13] Sebastian Arnold u. a. "Adaptive behavior with user modeling and storyboarding in serious games". In: *Proceedings - 2013 International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2013* (2013), S. 345–350. DOI: 10.1109/SITIS.2013.63.

[BA18] Franco D Berdun und Marcelo G Armentano. "Modeling Users Collaborative Behavior with a Serious Game". In: 1502.c (2018), S. 1–9. DOI: 10.1109/TG.2018.2794419.

[Bak+17] Abdellah Bakhouyi u. a. "Evolution of standardization and interoperability on E-learning systems: An overview". In: *2017 16th International Conference on Information Technology Based Higher Education and Training, ITHET 2017* (2017). DOI: 10.1109/ITHET.2017.8067789.

[Bie16] Christian Biegemeier. "Web-basierte Schnittstelle zur Analyse und Adaption von Serious Games". Master Thesis. KIT; Fraunhofer IOSB, 2016, S. 78.

[Che07] Jenova Chen. "Flow in games (and everything else)". In: *Communications of the ACM* 50.4 (2007), S. 31. ISSN: 00010782. DOI: 10.1145/1232743.1232769. URL: http://portal.acm.org/citation.cfm?doid=1232743.1232769.

[Csi88] Mihaly Csikszentmihalyi. "The flow experience and its significance for human psychology." In: *Optimal experience: Psychological studies of flow in consciousness*. New York, NY, US: Cambridge University Press, 1988, S. 15–35. ISBN: 0-521-34288-0 (Hardcover).

[Dör+16a] Ralf Dörner u. a. "Introduction". In: *Serious Games: Foundations, Concepts and Practice*. Hrsg. von Ralf Dörner u. a. Cham: Springer International Publishing, 2016, S. 1–34. ISBN: 978-3-319-40612-1. DOI: 10.1007/978-3-319-40612-1_1. URL: https://doi.org/10.1007/978-3-319-40612-1%7B%5C_%7D1.

[Dör+16b] Ralf Dörner u. a. *Serious Games - Foundations, Concepts and Practice*. 2016, S. 421. ISBN: 978-3-319-40611-4. DOI: 10.1007/978-3-319-40612-1. URL: http://link.springer.com/10.1007/978-3-319-40612-1.

[Dyc11] Sergius Dyck. "Benutzermodellierung beim arbeitsbegleitenden Lernen unter Berücksichtigung des Aufgabenkontextes". Master Thesis. Hochschule Karlsruhe - Technik und Wirtschaft, 2011, S. 126.

[Fie00] Roy Thomas Fielding. "Architectural Styles and the Design of Network-based Software Architectures". In: *Building* 54 (2000), S. 162. ISSN: 1098-6596. DOI: 10.1.1.91.2433. arXiv: arXiv:1011.1669v3. URL: http://www.ics.uci.edu/%7B~%7Dfielding/pubs/dissertation/top.htm.

[Fla11] David Flanagan. *JavaScript: The Definitive Guide 6th Edition*. 2011, S. 994. ISBN: 9780596805524. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3. URL: http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract.

[Gam+95] Erich Gamma u. a. *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN: 0-201-63361-2.

[GBB17] Zheng Gao, Christian Bird und Earl T. Barr. "To Type or Not to Type: Quantifying Detectable Bugs in JavaScript". In: *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017* (2017), S. 758–769. DOI: 10.1109/ICSE.2017.75.

[Goe07] Stefan Goessner. *JSONPath - XPath for JSON*. 2007. URL: http://goessner.net/articles/JsonPath/ (besucht am 21. 08. 2018).

[Ham04] Paul Hamill. *Unit Test Frameworks*. First. O'Reilly, 2004. ISBN: 0596006896.

[Hsi16] I-Han Hsiao. "Mobile Grading Paper-Based Programming Exams: Automatic Semantic Partial Credit Assignment Approach". In: *Adaptive and Adaptable Learning: 11th European Conference on Technology Enhanced Learning, EC-TEL 2016, Lyon, France, September 13-16, 2016, Proceedings Adaptable Learning* (2016), S. 110–123.

[JT17] Axel Jacob und Frank Teuteberg. "Game-Based Learning, Serious Games, 8 Business Games und Gamification – Lernförderliche Anwendungsszenarien, gewonnene Erkenntnisse und Handlungsempfehlungen". In: *Gamification and Serious Games*. Springer International Publishing, 2017, S. 97–112. DOI: 10.1007/978-3-658-16742-4_8.

[KP88] Glenn E Krasner und Stephen T Pope. "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80". In: *Joop Journal Of Object Oriented Programming* 1 (1988), S. 26–49. ISSN: 0896-8438. DOI: 10.1.1.47.366.

[KR16] Jonathan M Kevan und Paul R Ryan. “Experience API: Flexible, Decentralized and Activity-Centric Data Collection”. In: *Technology, Knowledge and Learning* 21.1 (Apr. 2016), S. 143–149. ISSN: 2211-1670. DOI: 10.1007/s10758-015-9260-x. URL: https://doi.org/10.1007/s10758-015-9260-x.

[Lei16] Sebastian Leidig. “Analysis of User Attention for Adaptive Serious Games - Design and Implementation of an Evaluation Framework”. Master Thesis. KIT; Fraunhofer IOSB, 2016.

[Len+16] B. Leng u. a. “Topic model based behaviour modeling and clustering analysis for wireless network users”. In: *2015 21st Asia-Pacific Conference on Communications, APCC 2015* (2016), S. 410–415. DOI: 10.1109/APCC.2015.7412547. arXiv: 1511.05618.

[Mic18] Microsoft. *TypeScript Handbook*. 2018. URL: https://www.typescriptlang.org/docs/handbook/basic-types.html.

[PST09] Dulyawit Prangchumpol, Siripun Sanguansintukul und Panjai Tantasanawong. “Server Virtualization by User Behaviour Model using a Data Mining Technique – A Preliminary Study”. In: *Management* (2009), S. 2–6.

[RC10] P Rashidi und D J Cook. “Mining Sensor Streams for Discovering Human Activity Patterns over Time”. In: *2010 IEEE International Conference on Data Mining* (2010), S. 431–440. ISSN: 2374-8486. DOI: 10.1109/ICDM.2010.40.

[RDS17] Jonathan Robie, Michael Dyck und Josh Spiegel. *XML Path Language (XPath) 3.1*. W3C Recommendation. W3C, 2017.

[Ris78] J Rissanen. “Modelling by the shortest data description”. In: *Automatica* 14 (1978), S. 465–471. ISSN: 00051098. DOI: 10.1016/0005-1098(78)90005-5.

[Rus18] Rustici Software. *The Enterprise Learning Ecosystem*. 2018. URL: https://xapi.com/ecosystem/ (besucht am 11. 08. 2018).

[Sof18] Rustici Software. *SCORM Explained*. 2018. URL: https://scorm.com/scorm-explained/ (besucht am 11. 08. 2018).

[SP17] James Snell und Evan Prodromou. *Activity Streams 2.0*. W3C Recommendation. W3C, 2017.

[SR17a] Alexander Streicher und Wolfgang Roller. “Adaptivity for Game Based Learning - Personalized Adaptive Learning for Military Image Interpretation”. In: *Fraunhofer IOSB Jahresbericht 2017* (2017), S. 64–65.

[SR17b] Alexander Streicher und Wolfgang Roller. “Interoperable Adaptivity and Learning Analytics for Serious Games in Image Interpretation”. In: *Data Driven Approaches in Digital Education: 12th European Conference on Technology Enhanced Learning, EC-TEL 2017, Proceedings*. Hrsg. von Lavoué É. u. a. Bd. 10474 LNCS. Tallinn: Springer International Publishing, 2017, S. 598–601. ISBN: 978-3-319-66610-5. DOI: 10.1007/978-3-319-66610-5_71. URL: https://link.springer.com/chapter/10.1007/978-3-319-66610-5%7B%5C_%7D71.

[SRB17] Alexander Streicher, Wolfgang Roller und Christian Biegemeier. "Application of Adaptive Game-based Learning in Image Interpretation". In: *Proceedings of the 11th European Conference on Games Based Learning, ECGBL 2017* (2017), S. 975–978. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85036467188%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=20b44f31543bf9ec02c1c2d08e5dc6ad.

[SS16] Alexander Streicher und Jan D Smeddinck. "Personalized and Adaptive Serious Games". In: *Entertainment Computing and Serious Games: International GI-Dagstuhl Seminar 15283, Dagstuhl Castle, Germany, July 5-10, 2015, Revised Selected Papers*. Hrsg. von R. Dörner et al. Cham: Springer International Publishing, 2016. Kap. 14, S. 332–377. ISBN: 978-3-319-46152-6. DOI: 10.1007/978-3-319-46152-6_14. URL: http://dx.doi.org/10.1007/978-3-319-46152-6%7B%5C_%7D14.

[Sti17] Stefan Stieglitz. "Enterprise Gamification - Vorgehen und Anwendung". In: *Gamification and Serious Games*. Springer Fachmedien Wiesbaden GmbH, 2017. Kap. I.1, S. 3–14. DOI: 10.1007/978-3-658-16742-4_1.

[Str17] Alexander Streicher. "Adaptive und adaptierbare Wissensvermittlung". In: *Intelligente bildgestützte Aufklärung in verteilten Sensorsystemen (IBAS)* (2017).

[SZ07] Valerie Shute und Diego Zapata-Rivera. "Adaptive technologies". In: *Handbook of research on educational communications and technology* 2007 (2007), S. 277–294.

[TPS16] Nutcha Temiyasathit, Proadpran Punyabukkana und Atiwong Suchato. "Course periodic behavior modelling and its application in LMS activity prediction". In: *IEEE Global Engineering Education Conference, EDUCON* 10-13-April-2016.April (2016), S. 1164–1174. ISSN: 21659567. DOI: 10.1109/EDUCON.2016.7474703.

[TV10] Stefan Tilkov und Steve Vinoski. "Node.js: Using JavaScript to build high-performance network programs". In: *IEEE Internet Computing* 14.6 (2010), S. 80–83. ISSN: 10897801. DOI: 10.1109/MIC.2010.145.

[Xie+17] Tao Xie u. a. "Modeling and Predicting the Active Video-Viewing Time in a Large-Scale E-Learning System". In: *IEEE Access* 5 (2017), S. 11490–11504. ISSN: 21693536. DOI: 10.1109/ACCESS.2017.2717858.

Abbildungsverzeichnis

KAPITEL A

Anhang

A.1. Konfigurierbarkeit der Logik von ElaiSim

Einstellung	Service	Beschreibung
Recording Port	Recording	Der Port, auf dem der Recording-Server gestartet wird.
Recording Methode	Recording	Die HTTP-Methode, auf der Push-Nachrichten des Serious Games erwartet werden (z.B. *POST*).
Recording Pfad	Recording	Der URI-Pfad, auf dem Push-Nachrichten erwartet werden.
Recording Parameter Name	Recording	Der Parameter, in dem in Push-Nachrichten das Statement verpackt ist.
Push URL	Playback	URL einschließlich Port, auf den Push-Nachrichten gesendet werden.
Push Methode	Playback	Die HTTP-Methode, mit der Push-Nachrichten gesendet werden (z.B. *POST*).
Push Parameter Name	Playback	Der Parameter, in dem das Statement bei Push-Nachrichten verpackt wird.
Pull URL	Playback	URL und Port, auf den Pull-Nachrichten gesendet werden.
Pull Methode	Playback	Die HTTP-Methode, mit der Pull-Nachrichten gesendet werden (z.B. *POST*).
Pull Payload	Playback	Der mit Pull-Nachrichten verknüpfte Datensatz, da die ELAI üblicherweise zusätzliche Informationen wie das Spiel-Genre bei Pull-Anfragen erwartet.
Pull Verzögerung	Playback	Die Verzögerung nach Abschließen einer Push-Nachricht, bevor eine Pull-Nachricht zum Abrufen der Performance-Daten gesendet wird.
Vergleichs-Epsilon	Verification	Eine Konstante ϵ, die beim Vergleich der empfangenen Performance-Werte mit den Referenzwerten als Spielraum verwendet wird.
Ignorierte Perf.-Werte	Verification	Eine Liste an Bezeichnern von Performance-Werten, die bei der Verifikation ignoriert werden sollen.

Tabelle A.1: Konfigurierbarkeit des Recording-, Playback- und Verification-Service von ElaiSim.