AxEL - Eine modulare Softwarekomponente für ein dediziertes E-Prüfungssystem zur Generierung von xAPI-Statements für Assessment Analytics

Martin Breuer[1], Annabell Brocker[2], Malte Persike[1] und Ulrik Schroeder[2]

Abstract: Um Lehrende zukünftig bei der Qualitätssicherung digitaler Prüfungen sowie der Optimierung von prüfungsbezogenen Lehr- und Lerninhalten zu unterstützen, ist die Entwicklung von Assessment Analytics Tools für das E-Prüfungssystem Dynexite geplant. Dieser Beitrag stellt die Entwicklung einer modularen Softwarekomponente vor, die bestehende Interaktions- und Ergebnisdaten von Studierenden in xAPI Statements umwandelt und diese der zentral an der Hochschule bereitgestellten Learning Analytics Infrastruktur zugänglich macht. Die vorgestellte Softwarekomponente und die geplante Weiterverarbeitung der Daten werden in das Learning Analytics Referenzmodell nach Chatti et. al. eingeordnet, um den Datenbedarf zu ermitteln. Die Softwarekomponente ermöglicht Untersuchungen auf unterschiedlichen Detailebenen. Für die Qualitätssicherung der Prüfungen ist meist eine genaue Zuordnung von feingranularen Interaktionsdaten auf Eingabemöglichkeiten von Interesse. Bei der diagnostischen Untersuchung von Ergebnisdaten kann hingegen auf Detailstrukturen verzichtet werden.

Keywords: E-Prüfungssystem, Assessment Analytics, Learning Analytics, xAPI, Prüfungsdaten.

1 Einleitung

Die Durchführung digitaler Prüfungen bietet für akademisches Personal technische, didaktische, rechtliche sowie organisatorische Chancen und Herausforderungen [Pe21]. Um diese zu adressieren, bildet das dedizierte E-Prüfungssystem Dynexite[3] verschiedene Phasen der Prüfung in einem einheitlichen Prozess ab. Allerdings fehlt bisher eine Assessment Analytics (AA) Komponente zur Auswertung der Prüfungsdaten mit dem Ziel der Optimierung von Lehr-, Lern- und Prüfungsinhalten.

Zum Schließen der Lücke werden die benötigten Daten im E-Prüfungssystem in xAPI Statements[4] umgewandelt und an die zentrale Learning Analytics (LA) Infrastruktur basierend auf Excalibur LA [JS22] gesendet. Innerhalb dieser Infrastruktur werden in Dynexite generierte Daten zunächst in einem Learning Record Store (LRS) abgelegt,

[1] RWTH Aachen, Center für Lehr- und Lernservices, {breuer@medien, persike@cls}.rwth-aachen.de, https://orcid.org/{0009-0008-0749-5110, 0000-0002-7825-089X}
[2] RWTH Aachen, Lerntechnologien, {a.brocker, schroeder}@cs.rwth-aachen.de, https://orcid.org/{0009-0007-6708-0892, 0000-0002-5178-8497}
[3] Dynexite Dokumentation, https://docs.dynexite.rwth-aachen.de, Stand: 27.03.2023
[4] xAPI-Spec, https://github.com/adlnet/xAPI-Spec, Stand: 27.03.2023

doi: 10.18420/delfi2023-17

dann mithilfe so genannter Analytics Engines ausgewertet und in einem Result Store gespeichert. Individuelle Einwilligungen werden durch eine Rights Engine in der zentralen LA Infrastruktur verwaltet, geprüft und gesichert, sodass nur berechtigte Personen Zugriff auf die erhobenen und ausgewerteten Daten haben. [JS22]

Dieser Beitrag beleuchtet die Frage, wie ein dediziertes E-Prüfungssystem an eine LA Infrastruktur zur kontinuierlichen Datenübertragung angeschlossen werden kann und wie Daten aus dem E-Prüfungssystem umgewandelt werden sollten. Das Ziel besteht darin, eine modulare Softwarekomponente für ein dediziertes E-Prüfungssystem zu entwickeln, um plattformübergreifende Analysen mittels xAPI Statements für AA zu ermöglichen.

2 Assessment Analytics im Hochschulkontext

Ellis definiert den Begriff AA als LA im Kontext von Assessments [El13]. LA ist zunächst das Messen, Sammeln, Analysieren und Auswerten von Daten über Lernende und ihren Kontext mit dem Ziel, das Lernen und die Lernumgebung zu verstehen und zu optimieren [SL11]. AA fokussiert die Definition auf den Kontext von elektronischen Assessments. Um zu entscheiden, in welcher Form Daten für eine AA Komponente benötigt werden, erfolgt eine Einordnung des Vorhabens in das LA Referenzmodell nach [Ch12] anhand der vier Dimensionen: Daten und Umgebungen (what?), Stakeholder (who?), Ziele (why?) und Methoden (how?).

2.1 Daten und Umgebungen (what?)

Dynexite ist eine an der RWTH Aachen entwickelte Prüfungssoftware, die als verteilte Infrastruktur konzipiert ist (vgl. Dynexite-Aufbau[5]). Die zentrale Plattformkomponente namens *Orbit* adressiert zwei Zielgruppen. Zum einen können Dozierende in *Orbit* Prüfungen erstellen und den Prüfungsprozess verwalten. Zum anderen können Studierende mithilfe von *Orbit* an Klausureinsichten teilnehmen sowie formative Self-Assessments mit automatischer Auswertung sowie untersemestrige Prüfungsleistungen (Hausübungen) absolvieren.

Der wesentliche von *Orbit* nicht abgedeckte Bereich sind die summativen Abschlussprüfungen. Für deren Durchführung dienen die *Satelliten* – speziell für die Durchführung von Abschlussprüfungen konzipierte Prüfungsserver. Sie werden in lokalen Netzwerken betrieben, in denen lediglich die Prüfungssoftware genutzt oder auf Anfrage von Dozierenden um weitere Tools erweitert werden kann. Auf externe Abhängigkeiten wird verzichtet, um Verzögerungen zu vermeiden. Vor der Durchführung einer Prüfung mit einem Satelliten müssen Prüfungen deshalb von *Orbit* übertragen werden. Dabei werden für die Durchführung der Prüfung notwendige Informationen migriert – für die nachfolgende Bewertung relevante Informationen

[5] Dynexite Aufbau, https://docs.dynexite.rwth-aachen.de/administration/structure, Stand: 26.03.2023

befinden sich ausschließlich im *Orbit*. Zur rechtssicheren Durchführung der Prüfungen werden Interaktionsdaten in Form von Events aufgezeichnet. Seit 2018 existiert ein Event-Log Datensatz mit über 53.000 Studierenden und 270 mio. Events (Stand 06/23).

2.2 Stakeholder (who?)

Die beiden primären Stakeholdergruppen für eine AA Anwendung sind Dozierende sowie Studierende. Statistische Auswertungskenntnisse seitens der Dozierenden können aufgrund der Heterogenität nicht vorausgesetzt werden. Sowohl Dozierende als auch Studierende könnten gleichermaßen von einer AA Anwendung profitieren, die auf mögliche Probleme bei Prüfungsaufgaben oder bei der Abstimmung von Prüfungs- auf Kursangebote hindeuten. Dabei liegt die Verhinderung von Datenmissbrauch im Interesse der Studierenden, da persönliche Daten nicht nur zur formativen Evaluation, sondern auch zur Prüfung und Bewertung verwendet werden könnten [Ch12].

2.3 Ziele (why?)

Gründe und Potenziale für den Einsatz einer AA Anwendung sind vielfältig. Schwierigkeitswerte können zur Optimierung des Aufgabenpools und als Basis für die Erstellung neuer Prüfungen dienen. Weitere Gütekriterien können herangezogen werden, um qualitativ unzureichende oder problematische Aufgaben zu identifizieren [SS22]. Neben der Erstellung der Prüfungen kann ein weiteres Ziel die Optimierung des prüfungsvorbereitenden Kontextes sein, indem erfolgskritische Übungsaufgaben fokussiert werden [SS22]. Ein weiteres Ziel ist die Weiterentwicklung bestehender Autorentools für Prüfungsaufgaben. Hier könnte die Aufgabenformulierung durch die Vorhersage von Gütekriterien anhand der Prüfungsdaten ähnlicher Aufgaben unterstützt werden. Über AA Ziele hinausgehend könnte akademisches Personal bei der Korrektur der Klausuren entlastet werden; eine Stakeholder Befragung diesbezüglich ist geplant.

2.4 Methoden (how?)

Zum aktuellen Zeitpunkt des Projekts, lässt sich eine Antwort auf die Frage „how?“ nur übergreifend skizzieren. Eine Übersicht von Auswertungs- und Anwendungs-möglichkeiten im Rahmen von AA sowie möglicher Integrationen in LMS ist in [SS22] zu finden. Die Methoden zur Erreichung der vorgenannten Ziele sind vielfältig. Für die geplante AA Anwendung liegt der Fokus auf Auswertungs- und Darstellungsmethoden, die Dozierende auf aufgabenbezogene Probleme mit dem Ziel der weiteren Untersuchung hinweisen. Erste Limitierungen und Möglichkeiten bei der Anwendung der Methoden lassen sich durch die Datenberücksichtigung unterschiedlicher Granularitäten ableiten. Die inhaltlichen Auswertungen der Prüfungsaufgaben erfordert keine Verarbeitung personenbezogener Studierendendaten, sodass ein anonymes Logging möglich ist. Bei zunehmender Granularität der Daten sind tiefergehende

Untersuchungen möglich, die Anforderungen an den Schutz der Daten hingegen wachsen, sodass je nach universitärem Kontext eine Konfiguration des Datenlogging wünschenswert oder notwendig ist.

3 Anforderung an eine Logging-Erweiterung

Bei der Entwicklung einer Logging-Erweiterung für das Prüfungssystem dienten bereits bestehende Logging-Systeme anderer Plattformen wie JupyterLab [Br22] und Moodle [JSS22] als Grundlage. Die ersten Herausforderungen ergeben sich beim Transfer von Daten aus dem Quellsystem Dynexite. Zwar wird die Laufzeit der Eventverarbeitung in anderen Systemen bereits durch einen Statement Puffer reduziert, die Datenumwandlung erfolgt jedoch direkt nach Eintreten des Ereignisses [JSS22]. Die verteilte Dynexite Architektur mit gegebenenfalls eingeschränkten Internetverbindungen, verbietet ein direktes Übertragen der Daten während der Prüfung. Zusätzlich werden kontextuelle Informationen, die nicht notwendig für die Bearbeitung einer Prüfung sind, wie z. B. Musterlösungen, Kurszuordnung usw., nicht auf die Prüfungsserver *Satellite* migriert. Ein vollständiges generieren der xAPI Statements (vgl. Abb. 1) mit Kontext und Erweiterungen ist somit während einer Prüfung nicht ohne weiteres möglich.

Ein einheitliches Übertragen der Daten kann ohne große Änderungen am Prüfungssystem nur über die zentrale SQL-Datenbank des *Orbits* im Backend erfolgen (vgl. Abb. 1). Insgesamt birgt ein vollständig asynchroner Ablauf der Verarbeitung und Übertragung der Daten am wenigsten Risiken. Die Versionierung der Daten in Dynexite erfolgt per so genanntem *Upcasting* beim Laden der Daten aus der Datenbank. Dies muss analog beim Logging erfolgen, um Kompatibilität mit historischen und zukünftigen Daten zu sichern. Hieraus ergibt sich Verwendung des gleichen Technologie-Stack mit Golang[6] als Programmiersprache und Docker[7] zum Deployment.

Zur Ermöglichung von Untersuchungen auf unterschiedlichen Detailebenen und zur Schnürung von Paketen für die Studierendenzustimmung, werden Statements auf drei Ebenen definiert. Diese ermöglichen eine flexible und detaillierte Erfassung von Prüfungsdaten. Auf der höchsten Ebene werden übergeordnete Prüfungsvorgänge erfasst, wie z. B. der Start oder das Ende einer Prüfung. Auf der zweiten Ebene werden Aktionen erfasst, die sich auf Fragen innerhalb der Prüfung beziehen, wie zum Beispiel das Öffnen, Beantworten oder Überspringen einer Frage. Diese Ebene erlaubt Analysen auf Basis der Zeitstempel zur Ermittlung der Bearbeitungsdauer einer Aufgabe und zur Abschätzung, wie viel Zeit künftig für die Bearbeitung eingeräumt werden sollte [SS22]. Auf der niedrigsten Ebene kann zusätzlich erfasst werden, auf welche Aufgabenteile sich die Statements beziehen. Ein Statement, das die Auswahl einer Antwortoption repräsentiert, kann zur Distraktoranalyse (vgl. [SS22]) verwendet werden. Zeitstempel erlauben auf dieser Ebene Untersuchungen des Rateverhaltens [SS22].

[6] Go, https://go.dev/, Stand: 27.03.2023
[7] Docker, https://www.docker.com/, Stand: 27.03.2023

4 AxEL: Asynchronous xAPI Event Logger

Die Herausforderungen bei der Umwandlung von Prüfungsdaten aus Dynexite in xAPI Statements werden durch die Implementierung eines sogenannten asynchronen xAPI Event Logger (AxEL) adressiert. AxEL besteht aus einem zentralen Service (*AxEL (Core)*) und drei modularen Komponenten (vgl. Abb. 1). Der *AttemptFinder* verwaltet, welche Events bereits umgewandelt wurden und bei welchem Versuch neue Events zur Umwandlung verfügbar sind. Die Events dieses Versuches werden vom *StatementGenerator* in xAPI Statements überführt. Der interne Zustand von Prüfungen und Versuchen ändert sich im Laufe der Zeit. Für die Umwandlung eines Events in ein xAPI Statement müssen daher alle vorherigen Events des Versuches berücksichtigt und in den Gesamtzustand integriert werden, sodass die Reproduzierbarkeit der Statement-Generierung möglich ist. Die Authentifizierung beim LRS und die Übertragung der Statements über die API des LRS übernimmt der *XAPIAdapter*. Diese Services werden in AxEL geladen und können flexibel je nach Kontext (z. B. Produktion oder Testumgebung) ausgetauscht werden. AxEL übernimmt zudem die Fehlerbehandlung, Queueing und Batching des Statement-Versandes sowie zu welchem Zeitpunkt ein Logging stattfinden soll, z. B. immer um drei Uhr morgens.

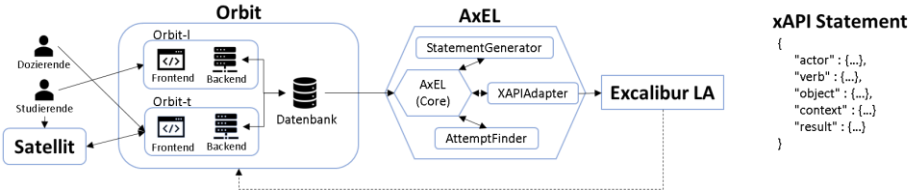

Abb. 1: Softwarearchitektur. Excalibur LA siehe [JS22]. xAPI Statement-Struktur (rechts).

In einem xAPI Statement sind bestimmte Angaben verpflichtend, wie zum Beispiel der *Aktor*, der die Aktion ausführt, das *Verb*, das die Aktion beschreibt, und das *Objekt*, das die Aktivität beschreibt. Darüber hinaus können optionale Felder hinzugefügt werden, die den *Kontext* und das *Ergebnis* beschreiben. Auf der obersten Ebene beziehen sich die Statements jeweils auf ein Assessment, sodass das Objekt eindeutig eine Prüfung (Blueprint) referenziert und mithilfe von *Erweiterungen* Basisinformationen zum Typ des Assessments (überwachte Prüfung, Hausübung oder Übung), Bearbeitungszeitraum, möglicher Bearbeitungszeit sowie bei Prüfungen Informationen zur Kohorte beinhaltet. Im Kontext befinden sich Informationen zum Prüfungssystem, Kurs sowie Organisationseinheit. Die Verben *started*, *stopped* und *graded* geben Aufschluss über Start und Ende einer Prüfung sowie im Falle eines *graded*-Statements, wie viele Punkte erreicht wurden. Durch weitere, über AxEL hinausgehende Maßnahmen, wird der Schutz der Dozierenden gewährleistet. Analog werden auf den tieferen Ebenen jeweils Aufgabenteile referenziert.

5 Fazit und Ausblick

In diesem Beitrag wurde eine Erweiterung für das Prüfungssystem Dynexite vorgestellt, die interne Event-Daten in xAPI Statements überführt und in die zentrale Learning Analytics Infrastruktur basierend auf [JS22] überträgt. Die Definition der xAPI Statements auf unterschiedlichen Detailebenen ermöglicht die Zusammenstellung von Paketen für die Zustimmung der Studierenden, so können Studierende beispielsweise nur generellen Statistiken oder auch detaillierteren Auswertungen des Antwortverhaltens zustimmen. Sowohl ein reines Berücksichtigen von Ergebnisdaten, als auch ein kleinschrittiges Verfolgen der Prüfungsbearbeitung ist möglich. Durch die vollständig asynchrone Strategie der Erweiterung, wird der Prüfungsprozess nicht beeinflusst.

In zukünftigen Versionen des Prüfungssystems wird eine Zuordnung von Aufgabenteilen erleichtert. Dies vereinfacht die hier vorgestellte Datenübertragung und ermöglicht die Entwicklung konkreter AA Komponenten, wie einer Distraktoranalyse, die genaue Referenzierung der Detailstrukturen benötigt [SS22]. Im Rahmen des Projektes NOVA:ea, gefördert durch die Stiftung Innovation in der Hochschullehre, wird ein Prüfungscockpits zur iterativen Verbesserung der E-Prüfungen entwickelt.

Literaturverzeichnis

[Br22] Brocker, A. et.al.: Juxl: JupyterLab xAPI Logging Interface. In: 2022 International Conference on Advanced Learning Technologies (ICALT), S. 158–160, 2022.

[Ch12] Chatti, M.A. et.al.: A Reference Model for Learning Analytics. In: International Journal of Technology Enhanced Learning (IJTEL) – Special Issue on “State-of-the-Art in TEL”, S. 318–331, 2012.

[El13] Ellis, C.: Broadening the scope and increasing the usefulness of learning analytics: The case for assessment analytics. In: British Journal of Educational Technology, S. 662-664, 2013.

[JSS22] Judel, S.; Schnell, E.; Schroeder, U.: Performantes xAPI Logging in Moodle. In: 20. Fachtagung Bildungstechnologien (DELFI), S. 159–164, 2022.

[JS22] Judel, S.; Schroeder, U.: EXCALIBUR LA - An Extendable and Scalable Infrastructure Build for Learning Analytics. In: 2022 International Conference on Advanced Learning Technologies (ICALT), S. 155–157, 2022.

[Pe21] Persike, M. et.al.: Digitale Prüfungspraxis: Szenarien, Perspektiven, Empfehlungen. In: Whitepaper - Digitale Prüfungen in der Hochschule, 2021.

[SS22] Scheidig, F.; Schweinberger, K.: Assessment Analytics - Daten digitaler Prüfungen auswerten. In: Neues Handbuch Hochschullehre (NHHL), Ausgabe 108, 2022.

[SL11] Siemens, G.; Long, P.: Penetrating the fog: Analytics in learning and education. In: EDUCAUSE review 46(5), S. 31-40, 2011.