Modularisierung als Schlüssel zur Skalierung

Erfahrungen aus der Entwicklung des E-Assessment-Systems JACK

Lukas Glaser[1] und Michael Striewe [2]

Abstract: Bildungstechnologische Systeme, die in zeitlich begrenzten Projekten, konkreten Lehrveranstaltungen oder individuellen Qualifikationsvorhaben entwickelt werden, sind nicht immer für den universellen Einsatz geeignet. Soll der Einsatz dennoch deutlich über den ursprünglichen Anwendungsbereich hinaus skaliert werden, sind daher möglicherweise grundlegende Änderungen an der Software notwendig. Anhand von Erfahrungen aus der Entwicklung des E-Assessment-Systems JACK diskutiert der Beitrag, wie die systematische Umsetzung von Modularität in der fachlichen Architektur und dem Systemdesign die Skalierbarkeit eines Systems verbessern kann.

Keywords: E-Assessment, Skalierbarkeit, Modularität, Softwarearchitektur, Komponenten

1 Einleitung

Bildungstechnologische Systeme kommen in zahlreichen Bildungsinstitutionen zum Einsatz und spielen dort teilweise eine zentrale Rolle. Neben etablierten Systemen, die als fertige Softwareprodukte eingesetzt werden können, sind auch Systeme im Einsatz, die Gegenstand kontinuierlicher Forschung und Weiterentwicklung sind. Solche Systeme, die vielleicht initial im Rahmen von zeitlich begrenzten Projekten, konkreten Lehrveranstaltungen oder individuellen Qualifikationsvorhaben entwickelt worden sind, sind jedoch nicht notwendigerweise für den universellen Einsatz geeignet. Soll der Einsatz dennoch deutlich über den ursprünglichen Anwendungsbereich hinaus skaliert werden, sind daher möglicherweise grundlegende Änderungen an der Software notwendig. Ungünstige Designentscheidungen in einer frühen Phase der Entstehung der Software können diese Änderungen aufwändig machen, was die Skalierung möglicherweise ganz verhindert.

Anhand von Erfahrungen aus der Entwicklung des E-Assessment-Systems JACK diskutiert dieser Beitrag, wie die systematische Nutzung von Modularität in der fachlichen Architektur und dem Systemdesign die Skalierbarkeit eines Systems verbessert und zur Sicherstellung seines langfristigen Erfolgs beiträgt. Der Beitrag richtet sich primär an

[1] Universität Duisburg-Essen, paluno – The Ruhr Institute for Software Technology, Gerlingstraße 16, 45127 Essen, lukas.glaser@paluno.uni-due.de
[2] Universität Duisburg-Essen, paluno – The Ruhr Institute for Software Technology, Gerlingstraße 16, 45127 Essen, michael.striewe@paluno.uni-due.de, https://orcid.org/0000-0001-8866-6971

doi: 10.18420/wsdelfi2023-33

diejenigen, die beim Entwurf eines Systems vor grundlegenden Designentscheidungen stehen und die Perspektive einer langfristigen Nutzung im Auge behalten wollen.

1.1 Verwandte Arbeiten und State-of-the-Art

Unter den Stichworten „Software Evolution“ [Re19] und „Long-living Software Systems“ [Du12] befasst sich die Forschung im Software Engineering mit dem Phänomen, dass grundlegende Änderungen an einer Software nötig sein können, um diese langfristig nutzen zu können. Fallstudien (z. B. [ZKH18]) zeigen, dass ein Umbau monolithischer Software hin zu einer modularen Architektur hilfreich oder sogar notwendig sein kann. Fallstudien aus der Bildungstechnologie (wie z. B. [SZG15], [St23a]) sind aber selten.

Modularität und Skalierbarkeit von Bildungstechnologie werden in der Literatur dagegen durchaus regelmäßig diskutiert (z. B. [Rö21], [Zs18], [Hi21]). Meist geht es in diesen Fällen jedoch nicht um das Verlassen des ursprünglichen Anwendungskontexts. Bußler et al. [Bu21] befassen sich mit der Langlebigkeit von Bildungstechnologie über das Projektende hinaus, fokussieren dabei aber insbesondere die IT-Infrastruktur und weniger die fachliche Architektur und das Systemdesign der Systeme.

2 Designentscheidungen zur Skalierung

2.1 Modularität der Inhalte

Zentrale Elemente zur Organisation von Inhalten in einem E-Assessment-System sind häufig *Aufgaben* als kleinste inhaltliche Einheiten, die von Lehrenden angelegt und von Lernenden bearbeitet werden. Im Laufe der Entwicklung und des Einsatzes von JACK wurden analog dazu zunächst auch zahlreiche verschiedene Aufgabentypen entwickelt. Dabei entstand in einigen Aufgabentypen der Wunsch, diese in Teilaufgaben unterteilen zu können. Um die technische Komplexität beherrschbar zu halten, erwies es sich als vorteilhaft, *Teilaufgaben* als kleinste inhaltliche und technische Einheit zu betrachten und beliebige Kombinationen von Teilaufgabentypen zuzulassen, anstatt nur in bestimmten Aufgabentypen bestimmte Kombinationen zu ermöglichen.

Jede Aufgabe ist demnach bereits modular und besteht aus einer oder mehreren Teilaufgaben. Jeder Teilaufgabentyp ist ein in sich geschlossenes Modul mit einer gleichförmigen Schnittstelle nach außen, das jeweils eigene Datenmodelle, GUIs und Geschäftslogik implementiert. Das Datenmodell definiert, wie Aufgabendefinitionen und Einreichungen persistiert werden. Die GUIs formen grafische Schnittstellen zur Aufgabendefinition (z. B. die Definition von Eingabefeldern) und zur Einreichung (z. B. über einen Formeleditor). In der Geschäftslogik wird das Bewerten von Einreichungen gesteuert und Feedback generiert (z. B. anhand einer Feedbackregel für ein bestimmtes Eingabefeld).

Technisch wird die Modularisierung über Vererbung und abstrakte Oberklassen sowohl für die Aufgabendefinition als auch die Einreichungen gelöst. Derzeit werden 11 verschiedene Module angeboten, darunter generische Typen wie Multiple-Choice und spezielle Typen für Programmier- und Modellieraufgaben (z. B. Java und UML). Dabei kann ein Modul beliebig komplex sein; so wird beispielsweise bei Programmieraufgaben der eingegebene Code an Microservices für statische Analysen und zum Testen gegen Testfälle gesendet (siehe Abschnitt 2.4). Durch die gleichförmige Schnittstelle aller Typen von Aufgabenteilen stehen generische Eigenschaften wie z. B. die Einstellung des Punktegewichts in Bezug auf die gesamte Aufgabe oder Funktionen wie beispielsweise die Möglichkeit, Hinweise anzufordern, automatisch in allen Typen zur Verfügung.

Organisatorisch werden Aufgaben in Ordnern abgelegt, die jedoch nur eine organisatorische und keine inhaltliche Funktion haben. Als übergeordnetes Konzept zur Zusammenfassung von Aufgaben dienen stattdessen *Kurse*. In diese können Aufgaben entweder fest oder basierend auf einer Auswahl aus Ordnern zugewiesen werden. Dadurch wird die Wiederverwendung von Aufgaben in mehreren Kursen ermöglicht.

2.2 Modularität von Präsentation und Zugriff

Die Struktur aus Teilaufgaben, Aufgaben und Kursen ist für den Zugriff aber noch nicht ausreichend modular. Würden Einstellungen für den Zugriff direkt im Kurs getroffen, kann derselbe Kurs nicht in verschiedenen Kontexten wiederverwendet werden. Es gibt aber Situationen, bei denen dies erforderlich ist, z. B. bei einem Kurs, der in mehreren Klausuren verwendet wird, die unterschiedliche Einstellungen (z. B. Zeitlimits) haben.

Aus diesem Grund ist auch die Präsentation für Lernende in JACK modularisiert ausgelegt. Mit den sogenannten *Kursangeboten* wird eine zusätzliche Ebene über Kursen geschaffen, die den Zugriff regelt. Ein Kursangebot besteht aus drei konfigurierbaren Modulen: Das *Anmeldungsmodul* bietet typische Features, wie die Begrenzung von Plätzen, eine Warteliste und die Auswahl zwischen Terminen. Über das *Bearbeitungsmodul* wird ein Kursangebot mit einem Kurs verknüpft. Für unterschiedliche Einsatzzwecke kann die Bearbeitungssicht detailliert konfiguriert werden, u. a. kann ein Zeitlimit zugeschaltet werden, Feedback oder die Anzeige von Punktzahlen komplett deaktiviert werden oder das Wiederholen des Kurses erlaubt werden. Das *Reviewmodul* regelt den Zugriff nach Beenden des Kurses. Wird es zugeschaltet, können sich Lernende je nach Einstellung ihre Eingaben, erreichte Punktzahlen und Feedback anzeigen lassen.

Mit den drei Modulen sind beliebige Kombinationen möglich, die verschiedenen individuellen Anforderungen von Lehrenden Rechnung tragen. Das alleinige Nutzen des Anmeldungsmoduls wird beispielsweise für die Termineintragung zu mündlichen Prüfungen genutzt. Das Bearbeitungsmodul alleine wird ohne Beschränkungen wie Zeitlimits häufig für offene Übungskurse verwendet. Alle drei Module zusammen mit zeitlichen Beschränkungen werden dafür genutzt, dass sich Lernende zunächst für eine

Prüfung anmelden, erst zu einem späteren Zeitpunkt (ggf. in Präsenz) den Kurs bearbeiten und nach der Abgabe direkt ihre Ergebnisse sehen können.

2.3 Modularität von Rollen und Rechten

JACK wird sowohl für formative Assessments (z. B. Übungsaufgaben zur freien Bearbeitung) als auch für summative Assessments (z. B. in Klausuren) verwendet. Letzteres stellt besondere Anforderungen an das System, da Daten über Prüfungen und Prüflinge einem besonderen Schutz unterliegen [FH12]: Zu jedem Zeitpunkt muss sichergestellt werden, dass kein unbefugter Zugriff auf Prüfungsdaten stattfindet; außerdem sollen sensible Informationen nur einem kleinstmöglichen Personenkreis zugänglich sein. Ein typischer Anwendungsfall ist das Erstellen oder Testen von Aufgaben durch studentische Hilfskräfte. Diese dürfen zwar auf die Konfiguration der Aufgaben zugreifen, jedoch nicht auf Einreichungen, die im Kontext einer Prüfung angefallen sind.

Skaliert man ein System über den Kontext eines einzelnen Lehrstuhls hinaus, ergeben sich zwei Herausforderungen: Erstens müssen Lernende in der Lage sein, als Hilfskraft an einem Lehrstuhl Prüfungsinhalte zu erstellen, während sie gleichzeitig an anderer Stelle als „normale“ Lernende an Übungen und Prüfungen teilnehmen. Zweitens sind Aufgabenverteilungen und daraus resultierende Berechtigungen nicht bei allen nutzenden Lehrstühlen identisch, so dass es keine einheitliche Definition von Rollen oder Berechtigungsleveln geben kann.

JACK implementiert deshalb ein fein-granulares Rechtesystem: Es existieren fünf partiell unabhängige Aspekte (Lesen, Erweitertes Lesen, Schreiben, Bewerten, Verwalten), die jeweils bestimmte Ansichten und Aktionen freischalten. Pro Ordner kann jedes Recht individuell an Accounts oder Accountgruppen vergeben werden. Für einen Account ohne besondere Rechte auf einem Objekt gilt die Sicht für Lernende.

Technisch werden vergebene Rechte als gesetzte Bits gespeichert. Diese Lösung ist nicht nur schlank in der Datenspeicherung, sondern ermöglicht auch eine flexible Erweiterung, falls ein weiterer Aspekt durch ein eigenes Recht abgebildet werden soll. Im Sinne von Access Control Lists werden die vergebenen Rechte pro Ordner gespeichert. Alternativ könnten Rechte auch pro Account gespeichert werden. Typischerweise werden jedoch für jeden Ordner Rechte definiert sein, während viele Accounts keine Rechte haben werden, so dass der gewählte Weg effizienter ist.

2.4 Modularität der Systemarchitektur

Schon die erste Version von JACK enthielt eine modulare Systemarchitektur, um langlaufende Vorgänge bei der Bewertung von Programmieraufgaben als asynchrone Prozesse in separate Komponenten auslagern zu können und die Sicherheit des Systems zu erhöhen [St16]. Daraus ergibt sich bereits eine Skalierbarkeit bei steigender Last, indem zusätzliche Instanzen der benötigten Komponenten gestartet werden können, um den

Durchsatz des Systems zu erhöhen. Gleichzeitig erleichtert dies die Wartung des Systems bei steigender Nutzung: Je mehr unterschiedliche Nutzungen auf dem System stattfinden, umso unwahrscheinlicher wird es, ein geeignetes Zeitfenster zu finden, in dem das System zur Wartung heruntergefahren werden kann. Separate Komponenten können dagegen unabhängig vom übrigen Systembetrieb ausgetauscht werden [Dr17]. Daher wurden in jüngeren Versionen von JACK auch synchrone Prozesse auf die Nutzung von Microservices umgestellt und in separate Komponenten ausgelagert. Diese können nun ebenfalls unabhängig vom Kernsystem aktualisiert und bei hoher Last repliziert werden.

Neben der Skalierung in Bezug auf Last und Nutzungsumfang hilft diese Modularität auch bei der Skalierung des Entwicklungsprozesses und des Entwicklungsteams. Von allen Microservices laufen Instanzen im internen Netzwerk des Entwicklungsteams und müssen daher nicht auf jedem Entwicklungsrechner betrieben werden. Viele Microservices sind für die meisten Anwendungsszenarien zudem optional (da sie z. B. lediglich die Bewertung in einem bestimmten Aufgabentyp vornehmen), sodass selbst ohne Verfügbarkeit dieser Microservices sinnvoll an dem System gearbeitet werden kann. Umgekehrt können neue Features (z. B. neue Bewertungsverfahren) erst einmal unabhängig vom Gesamtsystem und ohne Rücksicht auf dessen Technologiestack entwickelt werden, um später über die vergleichsweise schmale und damit leicht zu realisierende Microservice-Schnittstelle angebunden zu werden [Th15]. Diese Art des Arbeitens ermöglicht es auch, Studierende im Rahmen von Studienarbeiten in die Entwicklung einzubeziehen, ohne dass dabei ein unverhältnismäßig hoher Einarbeitungs- und Betreuungsaufwand anfällt.

3 Diskussion und Fazit

Die vorgestellten Designentscheidungen betreffen Modularität und Skalierung an verschiedenen Stellen der fachlichen Architektur und des Systemdesigns. In nahezu allen Fällen ergaben sich die Entscheidungen aus Anforderungen, die erst durch den langfristigen und wachsenden Einsatz entstanden. Umgekehrt hat sich das Fehlen von Modularität als Hindernis für die weitere Entwicklung des Systems erwiesen, das – wenn überhaupt – nur mit hohem Aufwand überwindbar war. Die Erstellung eines modularen Systemdesigns war jedoch ebenfalls mit großem Aufwand verbunden. Teilweise haben sich Entscheidungen zur Modularisierung rückblickend sogar als falsch erwiesen und mussten überarbeitet werden [St23a]. Dieser jeweils einmalige Aufwand erscheint jedoch gerechtfertigt, da durch die Modularität auch komplexe Erweiterungen (z. B. [PS19], [St23b]) mit vergleichsweise geringem Aufwand umgesetzt werden konnten.

Literaturverzeichnis

[Bu21] Bußler, D.; Lucke, U.; Strickroth, S.; Weihmann, L.: Managing the Transition of Educational Technology from a Research Project to Productive Use. In: Proceedings of the Software Engineering 2021 Satellite Events, 2021.

[Dr17] Dragoni, N.; Giallorenzo, S.; Lafuente, A. L.; Mazzara, M.; Montesi, F.; Mustafin, R.; Safina, L.: Microservices: Yesterday, Today and Tomorrow. In: Present and Ulterior Software Engineering, Springer, Cham. S. 195-216, 2017. https://doi.org/10.1007/978-3-319-67425-4_12

[Du12] Durdik, Z.; Klatt, B.; Koziolek, H.; Krogmann, K.; Stammel J.; Weiss, R.: Sustainability guidelines for long-living software systems. In: 28th Int. Conf. on Software Maintenance (ICSM), Trento, Italy. S. 517-526, 2012. https://doi.org/10.1109/ICSM.2012.6405316

[FH12] Franke, P.; Handke, J.: E-Assessment. In: E-Learning, E-Teaching und E-Assessment in der Hochschullehre, Oldenbourg Wissenschaftsverlag, Kapitel 6, 2012. https://doi.org/10.1524/9783486716849.147

[Hi21] Hinrichs, T.; Bureau, H.; von Pilgrim, J.; Schmolitzky, A.: A Scaleable Online Programming Platform for Software Engineering Education. In: Proceedings of the Software Engineering 2021 Satellite Events, 2021.

[PS19] Pobel, S.; Striewe, M.: Domain-Specific Extensions for an E-Assessment System. In: Advances in Web-Based Learning - ICWL 2019, Springer, Cham. S. 327-331, 2019. http://doi.org/10.1007/978-3-030-35758-0_32

[Re19] Reussner, R.; Goedicke, M.; Hasselbring, W.; Vogel-Heuser, B.; Keim, J.; Märtin, L. (Hrsg.): Managed Software Evolution, Springer, 2019. https://doi.org/10.1007/978-3-030-13499-0

[Rö21] Röpke, R.; Drury, V.; Schroeder, U.; Meyer, U.: A Modular Architecture for Personalized Learning Content in Anti-Phishing Learning Games. In: Proceedings of the Software Engineering 2021 Satellite Events, 2021.

[St16] Striewe, M.: An architecture for modular grading and feedback generation for complex exercises. Science of Computer Programming, Volume 129, S. 35-47, 2016. http://doi.org/10.1016/j.scico.2016.02.009

[St23a] Striewe, M.: Architectural revision of the e-assessment system JACK. In: Companion Proc. 16th Europ. Conf. on Software Architecture (ECSA 2022) Tracks and Workshops, Prague, Czech Republic. 2023. https://doi.org/10.1007/978-3-031-36889-9_3

[St23b] Striewe, M.: Strukturformeln für Moleküle zeichnen und differenziertes Feedback erhalten: Eine integrierte Lösung im Rahmen des E-Assessment-Systems JACK. In: 21. Fachtagung Bildungstechnologien (DELFI), Aachen, S. 273-274, 2023. https://doi.org/10.18420/delfi2023-52

[SZG15] Striewe, M.; Zurmaar, B.; Goedicke, M.: Evolution of the E-Assessment Framework JACK. In: Gemeinsamer Tagungsband der Workshops der Tagung Software Engineering 2015, Dresden, Germany. S. 118-120, 2015.

[Th15] Thönes, J.: Microservices. IEEE Software, 32(1), S. 113-116, 2015. https://doi.org/10.1109/MS.2015.11

[ZKH18] Zirkelbach, C.; Krause, A.; Hasselbring, W.: On the Modernization of ExplorViz towards a Microservice Architecture. In: Combined Proceedings of the Workshops of the German Software Engineering Conference 2018. S. 39-42, 2018.

[Zs18] Zschaler, S.; White, S.; Hodgetts, K; Chapman, M.: Modularity for Automated Assessment: A Design-Space Exploration. In: Combined Proceedings of the Workshops of the German Software Engineering Conference 2018. S. 57-61, 2018.