Carsten Ullrich, Martin Wessner (Eds.): Proceedings of DeLFI and GMW Workshops 2017
Chemnitz, Germany, September 5, 2017

Audience Response Systems as an Example for Cross-University Cooperation

Alexander Kiy und Sven Strickroth

Abstract: Audience Response Systems (ARS) are an enhancement to teaching in higher education with the aim to strengthen participant activation and the involvement of students directly in lectures. There is a wide range of different solutions available. However, these solutions either rely on personal devices of students (so-called software clicker) or require the purchase of mostly commercial hardware solutions. Here, the presented Hands.UP approach tries to bridge these two complementary concepts. Based on a cost and effort estimation of selected ARS, the necessity of inter-university cooperation with regard to adequate further development and the use of ARS in teaching are motivated.

Carsten Ullrich, Martin Wessner (Eds.): Proceedings of DeLFI and GMW Workshops 2017
Chemnitz, Germany, September 5, 2017

Potentiale aufzeigen und Synergien nutzen: Audience Response Systeme als ein Anwendungsgebiet hochschulübergreifender Kooperationen

Alexander Kiy[1] und Sven Strickroth[1]

Abstract: Audience Response Systeme (ARS) stellen eine Ergänzung der Hochschullehre dar, um die Teilnehmeraktivierung zu stärken und die Studierenden unmittelbar in das Vorlesungsgeschehen einzubinden. Es existiert eine Fülle an Lösungen, die entweder ohne dedizierte Hardware auskommen (sogenannte Software-Clicker) oder die Anschaffung meist kommerzieller Hardware-Lösungen voraussetzen. An dieser Stelle versucht Hands.UP eine integrative Brücke zu schlagen. Auf Grundlage einer Kosten- und Aufwandsschätzung ausgewählter ARS-Lösungen soll die Notwendigkeit hochschulübergreifender Kooperationen hinsichtlich einer adäquate Weiterentwicklung und des Einsatzes von ARS in der Lehre motiviert werden.

Keywords: Audience Response System, Classroom Response System, Personal Response System, Clicker, Cost estimation, COCOMO

1 Einleitung

Audience Response Systeme (ARS), auch Classroom Response Systeme (CRS) oder „Clicker“ genannt, werden immer häufiger an Hochschulen eingesetzt. Dabei handelt es sich um E-Learning-Systeme, die in unterschiedlichsten didaktischen Szenarien (z. B. Wissensüberprüfungen, Feedback, Gruppeninteraktionen) zur Aktivierung von Studierenden oder zur Steigerung der Interaktivität (z. B. zur Auswahl von möglichen Vertiefungen) in großen Vorlesungen und Seminaren eingesetzt werden. Jedoch besteht nach wie vor der typische Einsatzzweck von ARS in der Durchführung einfacher Abstimmungen. Die verschiedenen Systeme ermöglichen eine automatisierte Auswertung und eine sofortige Bereitstellung der (anonymen, aggregierten) Ergebnisse ohne Zeitverlust. In den letzten Jahren fand eine rege Entwicklung solcher Systeme statt [Keo12, Kun13]. Folglich existieren unterschiedliche Systeme[2] [FB15] und empirische Ergebnisse, die den Nutzen von Clickern bestätigen [Mor09, OK13]. Neben einigen bundesweit bekannten und auch kommerziell beworbenen Systemen gibt es nach wie vor eine „Dunkelziffer“ von

[1] Institut für Informatik und Computational Science, Universität Potsdam, August-Bebel-Str. 89, 14482 Potsdam, vorname.nachname@uni-potsdam.de

[2] Übersichten und Vergleiche finden sich z. B. unter: http://ep.elan-ev.de/wiki/Audience_Response und http://www.brandhofer.cc/audience-response-systeme/ (letzter Abruf 2017-06-23)

kleineren, lokal entwickelten Systemen, die oft sehr ähnliche Kernfunktionen aufweisen. Häufig gibt es jedoch deutliche Unterschiede bei der Herangehensweise, den verwendeten Technologien und dem Funktionsumfang. Dies resultiert in einer Vielzahl untereinander inkompatibler Insellösungen. Dabei können sich an einzelnen Hochschulen[3], wie der Universität Potsdam, auch unterschiedliche Systeme (Hands.UP, PINGO, Hardware, sowie mindestens zwei von Lehrstühlen entwickelte Lösungen) parallel im Einsatz befinden. Probleme von Insellösungen und lokalen Entwicklungen bestehen vor allem bei der Weiterentwicklung und der Sicherstellung des langfristigen Betriebs mitsamt Support.

Zunächst werden überblicksartig Clicker-Lösungen skizziert. Anschließend wird die an der Universität Potsdam entwickelte ARS-Lösung zur Integration von Software- und Hardware-Clickern vorgestellt. Eine vergleichende Aufwandsschätzung, der bisher erbrachten Arbeiten wird diskutiert und zur Motivation für eine hochschulübergreifende Kooperation genutzt, um auf diesem Weg Kosten zu sparen und eine gleichbleibend hohe Qualität, Usability sowie langfristige Verfügbarkeit sicherzustellen.

2 Vielfältige Clicker-Systeme

Audience Response Systeme können grundsätzlich in Hardware- und Software-Clicker eingeteilt werden. Bei Hardware-Clickern handelt es sich um „single-purpose“ Hardware, die ausschließlich für diese Aufgabe hergestellt werden und Eingaben in Form vorgegebener Tasten (z. B. 0-9, Ja/Nein) entgegennehmen. Einige Systeme bieten die Möglichkeit Text auf den Geräten anzuzeigen. Sie können kabelgebunden, fest in Hörsälen installiert sein oder aus mobilen Einheiten bestehen, die über Funk oder Infrarot mit einer Basisstation kommunizieren. Vor einem Einsatz müssen die Systeme zum Hörsaal transportiert und ausgeteilt werden oder sind von den Studierenden mitzubringen. Zu den bekanntesten Hardware-Clickern zählen ActiVote, Quizdom Q6 und Interactive Voting System (IVS).

Neben den Hardware-Clickern gibt es einen Trend hin zu Apps bzw. Webseiten, die basierend auf dem BYOD-Ansatz auf mobilen Geräten der Studierenden verwendet werden. Einige Systeme benötigen eine lokale Installation auf den Geräten der Studierenden (z. B. Socrative), andere bestehen aus einer Webseite, die im Browser aufgerufen wird (z. B. ARSNova über QR-Code). Vorteile der sogenannten Software-Clicker liegen in der einfachen funktionalen Erweiterbarkeit, einer guten Skalierung bei vielen Abstimmenden und es können prinzipiell auch Studierende teilnehmen, die nicht vor Ort sind. Jedoch benötigen solche Systeme eine stabile Verbindung zu einem Server (in der Regel über das Internet) und die Geräte der Studierenden laden möglicherweise zur Ablenkung ein. Der Betrieb von Software-Clickern kann entweder über eine lokale Installation (z. B. ARSNova), über die Bereitstellung eines externen Dienstleister (z. B. PINGO) oder über dedizierte Clicker-Plugins für Lern-Management-Systeme (z. B. Cliqr für Stud.IP) erfolgen. Um nicht nur auf eine Kategorie (Hardware oder Software) angewiesen zu sein,

[3] Übersicht über in NRW-Hochschulen genutzte Systeme: http://www.eassessmentnrw.de/einsatzorte-in-nrw/formative-assessments/feedbacksysteme.html (letzter Abruf 2017-06-23)

wurden hybride Frameworks entwickelt, die versuchen beide Welten auf technischer Ebene miteinander zu kombinieren (z. B. Hands.UP, siehe nächster Abschnitt).

Clicker lassen sich weiter bezüglich der Verfügbarkeit und der Kosten kategorisieren [FB15]. Hardware-Clicker sind, sofern nicht selbstgebaut, ausschließlich kommerziell erhältlich und benötigen eine zusätzliche Software, die zur Auswertung genutzt wird. Teilweise müssen Hardware-Dongles erworben werden, die eine zusätzliche Lizensierung der Abstimmungsgeräte erfordern. Im Extremfall erfolgt beim Kauf eine Bindung an einen Hersteller (Vendor Lock-in) und bei Erweiterungen sind nicht nur Geräte, sondern auch neue Basisstationen und Lizenzen zu erwerben. Jedoch sind meist keine jährlichen Kosten zu entrichten. Bei Software-Clickern kann zwischen kommerzieller Software (z. B. EduVote), Freeware (z. B. feedbackr.io) und Open Source (z. B. ARSNova) unterschieden werden: Erstere erfordern i. d. R. eine jährliche Gebühr, werden aber komplett wie Freeware-Angebote von Dritten gepflegt und betrieben. Bei den freien Angeboten besteht eine Abhängigkeit vom Anbieter, sodass der Dienst langfristig angeboten werden kann.

Wie bereits in der Einleitung angedeutet, besteht die Hauptfunktionalität der meisten Systeme auf dem Handling von Single-Choice-Fragen, wobei es teilweise spezielle Unterstützung für Ja/Nein-Fragen oder Likert-Skalen gibt [FB15]. Multiple-Choice-Fragen werden noch von der Mehrzahl der Systeme unterstützt. Bei der Darstellung gibt es bereits große Unterschiede. So existieren Software-Clicker, die wie Hardware-Clicker Buttons nur mit vorgegebenen Beschriftungen anzeigen (vgl. EduVote). Funktionen wie Live-Feedback zum Dozenten (z. B. anonym Fragen stellen, „Abgehängt“-Button), Spontanumfragen, simultane Unterstützung von Hardware- und Software-Lösungen oder Abbildung weiterer Fragenformate (z. B. nummerische und Freitext-Antworten, Lückentexte oder die Markierung von Bereichen auf Bildern) treten sehr fragmentiert auf. Es gibt kein System mit einer breiten Unterstützung dieser Features. Auch wenn einzelne Systeme sehr viele Funktionen bieten (z. B. AMCS und ARSNova), unterstützen viele Systeme lediglich die Hauptfunktionalität (Single-Choice) und fokussieren auf einige ausgewählte Features. Darüber hinaus, wurden viele Entwicklungen mit viel Aufwand begonnen, die zwar die typischen Umfrage-Anwendungsfälle unterstützen, jedoch diese nur rudimentär umsetzen. Überspitzt pointiert argumentiert gibt es viele Systeme, die „das gleiche“ machen, aber kein System, das einem Dozierenden erlaubt „aus dem Vollen zu schöpfen“.

3 Hands.UP: Ein Integrativer Ansatz

Das Clicker-System Hands.UP basiert auf der Integrationslösung Click2Vote [ZHL14] und verfolgt das Ziel eine webbasierte, offene Plattform für die systemübergreifende Verwendung unterschiedlicher ARS-Lösungen zu schaffen. Das System zeichnet sich durch einen Webservice aus, an den Software- und Hardware-Clicker angebunden werden können. Click2Vote bzw. die Weiterentwicklung Hands.UP stellt die Datendrehscheibe zwischen dem in der Lehre verbreiteten und verwendeten Lernmanagementsystem

Moodle, klassischen Präsentationswerkzeugen wie PowerPoint und eingesetzten Software- und Hardware-Clickern dar. Für die Anbindung von Hardware-Clickern wurde das ResponseCard-System von Turning Technologies integriert: Hauptbestandteile dieser Hardwarelösung sind eine beliebige Anzahl an Clicker-Clienten (Response Card RF) und eine Basisstation (RF Receiver) welche per USB an den Dozierendenrechner angebunden wird.[4] Mittels des vorhandenen Software Development Kits (SDK) ist es möglich, die abgegebenen Stimmen der Clients mit Hilfe eines Windows-Programms namens ClickerManager über den Webservice weiterzugeben.

Die bisherige Lösung bestand aus einer Webseite, einer nativen Android und iOS App, einem Moodle-Plugin auf Basis des „Questionnaire"-Plugins[5], einem PowerPoint-Plugin, einem ClickerManager und einem Webservice zur Integration von Software- und Hardware-Clickern (Architektur vgl. Abb. 1). Umfragen können sowohl über die Administrations-Oberfläche der Webseite als auch über das Moodle-Plugin angelegt werden. Für die Präsentation der Umfragen kann das Moodle-Plugin, die Präsentationsansicht der Webseite oder auch das C-basierte Office-AddIn für PowerPoint genutzt werden. Die Abstimmungen können sowohl über die Apps, die Webseite, das Moodle-Plugin als auch mit Hilfe der angebundenen Hardware-Clicker über den ClickerManager erfolgen.

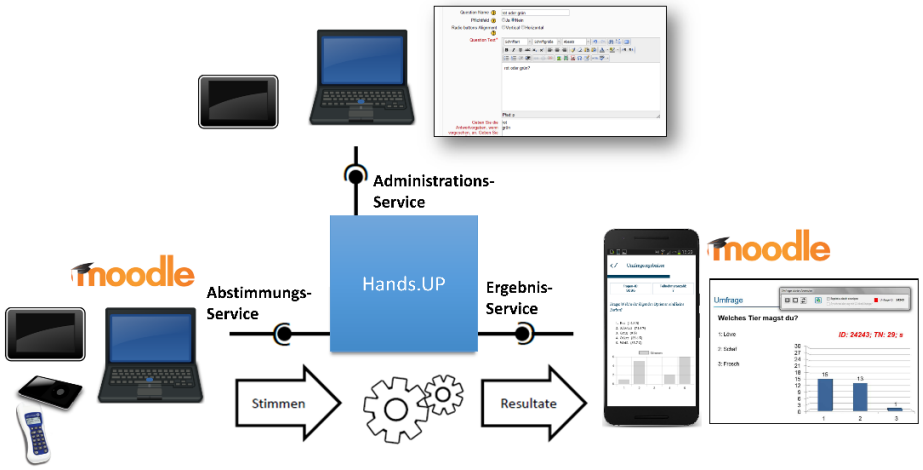

Abbildung 1: Übersicht Architektur/ und Integration der unterschiedlichen Plattformen

In Folge von Evaluationen (Nutzendentests) und Konsolidierungen mit anderen Applikationen mit dem Ziel Click2Vote in Form einer Weiterentwicklung namens Hands.UP für die Breite der Hochschule zugänglich zu machen, wurde das Webservice-Interface überarbeitet. Zur Erleichterung der Pflege der mobilen Applikationen wurde, vor dem Hintergrund eines vergleichsweise geringen Funktionsumfangs, eine hybride Implementierung der App gewählt. Bisher bestanden der Webservice und die Webseite separat. Zur

[4] https://www.turningtechnologies.com/response-solutions/responsecard-rf (letzter Abruf 2017-06-23)
[5] https://moodle.org/plugins/mod_questionnaire (letzter Abruf 2017-06-23)

Reduktion von Funktionsdoppelungen wurden beide Bestandteile in eine WebApp zusammengeführt und im gleichen Zug etablierte Frameworks (Hibernate, Spring etc.) eingesetzt. Studierenden und Dozierenden steht es nach wie vor offen über die Webseite, die App, Moodle oder mittels Hardware-Clickern abzustimmen. Die grundlegende service-orientierte Architektur blieb erhalten, lediglich die Anzahl der Einzelsysteme wurde reduziert und deren Wartbarkeit erhöht. Im Vergleich zu klassischen Systemen werden Systemgrenzen überwunden und ist unerheblich über welches System die Eingaben und Ergebnispräsentationen erfolgen.

4 Aufwandsschätzung

Das Projekt Click2Vote und die Weiterentwicklung hin zu Hands.UP wurden im Rahmen unterschiedlicher studentischer Arbeiten umgesetzt, hierzu zählen u. a. eine Bachelor- und Masterarbeit, eine Praxisaufgabe, ein Projekt und die Beschäftigung einer Hilfskraft im Rahmen von 7,5 Stunden für 10 Monate zwecks Überarbeitungen und der Überführung in den Produktivbetrieb. Die reinen Arbeitszeiten würden sich auf eine Summe von 12 Personenmonaten (PM) unter Berücksichtigung des ECTS-Aufwands (1 ECTS ~ 30h) belaufen. Hinzu kommen weitere Aufwände wie konzeptionelle Arbeiten, der Betreuungsaufwand, die Einrichtung und Konfiguration der Build- und Betriebsumgebungen (z. B. Installation von Software, Zertifikate, Dokumentationen), die Nutzung des CI-Designs oder der Betrieb und die Durchführung von Nutzendentests. Insofern bilden die 12 PM die erbrachten Arbeiten nur unzureichend ab, geben jedoch bereits eine grobe Aufwandsschätzung der Implementierungsarbeiten. Für eine bessere Aufwandsschätzung kann das sogenannte Constructive Cost Model II (COCOMO II) genutzt werden [Boe00]. COCOMO versucht mit Hilfe von Parametern eine Berechnungsgrundlage und somit Aufwandsabschätzung in Personenmonaten (PM) und Entwicklungszeit (Time to Develop (TDEV) in Monaten) in Abhängigkeit der geschätzten Codezeilen (Lines of Code (LOC) in Tausend - kurz KLOC) zu liefern. Nach dem COCOMO-Modell ergeben sich unter Verwendung der spezifischen Parameter für organische Projekte, zu denen insbesondere Kleinteams gehören und sich gut auf ARS-Implementierungen im Hochschulkontext anwenden lassen, die folgenden Berechnungsformeln:

$$PM = 2.04 \cdot KLOC^{1.05} \qquad \text{bzw.} \qquad TDEV = 2.5 \cdot PM^{0.38}$$

Ein weiterer Algorithmus namens git-hours[6] versucht auf Basis eines Git-Repositories die erbrachten Programmierstunden abzuschätzen. Dabei werden für Programmierer auf Basis von Commits/Beiträgen für das Git-Repository sogenannte Coding-Sessions identifiziert, um daraus schließlich die erbrachten Programmierstunden abzuleiten. Alle Commits, die innerhalb eines Zeitraums von zwei Stunden erfolgen, gehören zur gleichen Session; alles darüber hinaus wird zu einer separaten Session gezählt. Dieses Verfahren funktioniert

[6] https://github.com/kimmobrunfeldt/git-hours (letzter Abruf 2017-07-20)

jedoch ausschließlich mit Git-Repositorien und einer entsprechenden Commit-Historie. Um einen besseren Eindruck der erbrachten Implementierungsaufwände zu vergleichbaren Projekten zu erhalten, wurden die Ergebnisse von den verbreiteten Open-Source-Projekten PINGO[7] und ARSNova[8] in Verbindung gesetzt. Dabei wurden zunächst die Hauptsoftwaremodule identifiziert und anschließend ausgewertet (siehe Tabelle 1).

Tabelle 1: Aufwandsschätzungen auf Basis von COCOMO II und git-hours

Projekt	KLOC	PM	TDEV in Monaten	git-hours in Monaten
Hands.UP				
Moodle-Plugin	12,541	17,08	7,35	
iOS App	16,963	46,90	10,79	
Android App	9,347	25,09	8,51	
PowerPoint AddIn	*72,304*	*214,95*	*19,24*	
ClickerManager	12,396	33,74	9,52	
Webservice	19,951	55,61	11,51	
Webseite	95,067	286,51	21,46	
WebApp	*101,129*	*305,72*	*22,00*	
Hands.UP App (hybrid)	0,584	1,36	2,81	
ARSNova				
arsnova.click	113,454	172,48	17,70	17,50
arsnova-backend	26,839	75,93	12,96	12,36
arsnova-flashcards	8,956	23,98	8,36	8,79
arsnova-ppt-integration	*49,304*	*143,79*	*16,52*	1,59
arsnova-mobile	99,742	301,33	21,88	26,79
PINGO				
PINGOWebApp	35,748	51,30	11,16	-
Remote-Win	*578,196*	*953,59*	*33,89*	-
pingo-push	0,144	0,16	1,24	-

Die auf den KLOC basierenden Ergebnisse werden bei einigen Software-Modulen augenscheinlich massiv verzerrt. Das liegt bei genauerer Betrachtung entweder an Bibliotheken, die direkt im Repository eingebunden wurden, an automatisiert generiertem Quelltext oder es liegt Quelltext vor, der im Wesentlichen nur adaptiert wurde, und die real geschriebenen Codezeilen nur einen Bruchteil darstellen. So wurden für das Moodle-Plugin von Hands.UP, welches auf dem Moodle-Modul „mod_questionnaire“ mit ca. 12,5 KLOC basiert, lediglich 212 Zeilen angepasst, so dass sich im Endeffekt lediglich 1,8 Entwicklungsmonate ergeben. Werden bei der Hands.UP WebApp die Bibliotheken herausgerechnet, resultieren bei 13,2 KLOC schließlich nur noch 9,17 Entwicklungsmonate. Diese Beobachtung ist wohl auch auf die Webseite und den Webservice übertragbar. Das PowerPoint-Plugin besitzt viele externe DLLs, welche die Ergebnisse ebenfalls massiv

[7] https://github.com/PingoUPB (letzter Abruf 2017-07-20)
[8] https://github.com/thm-projects (letzter Abruf 2017-07-20)

verfälschen. Ein Indiz für die unzureichende Validität des COCOMO-Modells für Microsoft-spezifische Implementierungen findet sich im Vergleich von git-hours und COCOMO (vgl. arsnova-ppt-integration). Diese Fehler finden sich wahrscheinlich auch bei den Ergebnissen von ARSNova und PINGO wieder (vgl. Remote-Win und arsnova-ppt-integration). Der Vergleich der ARSNova-Ergebnisse von git-hours und COCOMO zeigt eine deutliche Korrelation der verschiedenen Algorithmen (unter Einschränkung des o. g. PowerPoint-Plugins). Weiterhin stellt die WebApp von Hands.UP lediglich eine Überarbeitung und Zusammenführung der Webseite und des Webservices dar und fällt somit weit geringer aus. Das PINGO-Repository beinhaltet nicht alle Vorgängerversionen, die realistische Entwicklungszeit ist somit wahrscheinlich weitaus höher einzuschätzen.

Die Schätzungen können maßgeblich durch die Extraktion extern entwickelten Codes, mit Hilfe der Durchführung von Experten- und Vergleichsschätzungen oder genauer inhaltlicher Quelltext-Analysen verfeinert werden. Sowohl das COCOMO-Modell als auch git-hours haben ihre Stärken und Schwächen. Beide blenden Aspekte wie Code-Metriken und Coding-Styles aus, liefern jedoch eine grobe Aufwandschätzungen aller notwendigen Aufgaben (nicht nur die bloße Programmierung). Unter Ausklammerung des AddIns und der ursprünglichen Webseite/Webservices, ergibt sich für das Projekt Hands.UP eine bereinigte Aufwandsschätzung von 42,6 Entwicklungsmonaten, für ARSNova 77,42 und für PINGO 46,29 Entwicklungsmonaten. Auch wenn die Wahrheit wahrscheinlich irgendwo zwischen den einzelnen Abschätzungen in Tabelle 1 liegt und bei Hands.UP wohl deutlich die erste Abschätzung von 12 Entwicklungsmonaten übersteigt, muss sich die Frage bei jeder Weiterentwicklung und Überführung in den Regelbetrieb gestellt werden, inwieweit die bisherigen und künftigen Investition überhaupt noch in einem Verhältnis zum Erwerb einer kommerziellen Lösung stehen (vgl. [KS13] für Kostenabschätzungen; z. B. ca. 2000-8000 € für 50 Hardware-Clicker bzw. ca. 3000 €/Jahr für einen externen Software-Clicker). Eine Kooperation mit anderen Hochschulen, die ähnliche Funktionalitäten implementieren und Herausforderungen hinsichtlich des Betriebs meistern müssen stellt vor diesem Hintergrund eine mögliche Alternative dar.

5 Diskussion und Ausblick

In diesem Paper wurden verschiedene Kategorisierungsmöglichkeiten für Clicker-Systeme vorgestellt. Dabei wurde deutlich, dass es große Übereinstimmungen bei den Kernfunktionalitäten existierender Systeme gibt. Anschließend wurde Hands.UP vorgestellt, welches auf Basis von Webservices die Integration von Software- und Hardware-Clickern ermöglicht. Die sich anschließenden Aufwandsschätzungen sollen als Kenngröße dienen über mögliche Kooperationen und Synergien bei der hochschulweiten Entwicklung von ARS nachzudenken. Wie bei Software-Entwicklungen üblich, dürfen neben den Kosten für die reine Entwicklung, die auch schon beträchtlich sein können, der Betrieb und die Wartung nicht vernachlässigt werden, die die Entwicklungskosten schnell

übersteigen können. Externe Dienste bieten zwar eine finanzielle Planungssicherheit, jedoch ist man vom Anbieter abhängig, da Dienste auch eingestellt werden können und nicht mehr verfügbar sind (evtl. mitsamt der dort gespeicherten Daten). Aber auch bei Hardware-Lösungen darf die Wartung nicht vernachlässigt werden: Hier fallen in regelmäßigen Abständen Reinigungen, Batteriewechsel und evtl. Reparaturen an. Wie in den vorherigen Abschnitten bereits gesehen, werden nicht nur viele Funktionen doppelt implementiert, sondern benötigen über die Zeit weitere Ressourcen, die sich meist schwer quantifizieren lassen und einige Personen in (Weiter)-Entwicklungen einbinden. Eine Kooperation und *eine* gemeinsame Software bieten indes viele Vorteile in Bezug auf Wartung und Weiterentwicklungen. Dennoch darf nicht vernachlässigt werden, dass viele Systeme im akademischen Umfeld auf Grund einer innovativen Idee bzw. Forschungsfrage entstanden sind. Neuentwicklungen und akademische Prototypen haben daher durchaus auch ihre Daseinsberechtigung. Jedoch können positiv evaluierte Features auch wieder in eine gemeinsame Software einfließen, so dass diese nicht nur exklusiv in einem einzigen Prototypen (möglicherweise „verstauben“), sondern auch einer breiten Öffentlichkeit zur Verfügung stehen.

Dies birgt grundsätzlich die Gefahr, ein System zu einer „eierlegenden Wollmilchsau“ ausbauen zu wollen. Diesem Problem kann begegnet werden, indem sich zum einen auf Kernfunktionalitäten verständigt wird, die sich bereits in vielen existierenden Systemen finden, und zum anderen eine modulare Architektur genutzt wird.

Literaturverzeichnis

[Boe00] Boehm, B. et al.: COCOMO II Model Definition Manual. Technical report. Ver. 2.1. Center for Software Engineering, USC, 1995.

[FB15] Frenger, F.; Bernhardt, S.: Abschlussbericht Audience- Response- Systeme an der JLU, 2015. http://www.uni-giessen.de/fbz/svc/hrz/org/mitarb/abt/3/Archiv/projekte/mob-abst-projekt-bericht

[Keo12] Keough, S. M.: Clickers in the Classroom: A Review and a Replication. Journal of Management Education, 36/6, S. 822–847, 2012.

[KS13] Krüger, M.; Schmees, M. (Hrsg.): E-Assessments in der Hochschullehre: Einführung, Positionen & Einsatzbeispiele, Psychologie und Gesellschaft, 2013.

[Kun13] Kundisch, D. et al.: Classroom Response Systems. Informatik-Spektrum, 36/4, S. 389–393, 2013.

[Mor09] Morin, D. et al.: The “Clicker” project: A scholarly approach to technology integration. In: Real learning opportunities at business school and beyond, S. 97–107, 2009.

[OK13] Oigara, J.; Keengwe, J.: Students’ perceptions of clickers as an instructional tool to promote active learning. J. Educ Inf Technol, 18/1, S. 15–28, 2013.

[ZHL14] Zender, R.; Haucke, P.; Lucke, U.: Click2Vote - Systematische Integration heterogener Clicker-Lösungen. In: Proc. DeLFI 2014, S. 163-168, 2014.