Alles nur Spielerei? Neue Ansätze für digitales spielbasiertes Lernen von Softwareprozessen

Jöran Pieper, Institute for Applied Computer Science, FH Stralsund

Joeran.Pieper@fh-stralsund.de

Zusammenfassung

Softwareprozesse beschreiben Ansätze für die Produktion und Evolution von Software. Sie gehören zu den Wissensgebieten des Software Engineering (SE), die sich weniger gut allein durch klassische Vorlesungen vermitteln lassen. Kursprojekte, die Vorlesungen häufig begleiten, werden durch akademische Rahmenbedingungen begrenzt.

Simulation und Digital Game-Based Learning (DGBL) werden große Potentiale bei der Erweiterung der Lernerfahrung über Vorlesungen und Kursprojekte hinaus zugesprochen. Sie können dazu genutzt werden, Einsicht in die Notwendigkeit von Softwareprozessen zu erzeugen und den Erfahrungshorizont von Studierenden des SE auf virtuelle und effiziente Art und Weise zu erweitern.

Verschiedene Anstrengungen unterschiedlicher Forschungsgruppen weltweit zeigen ermutigende Ergebnisse. Im Forschungsprojekt *Sim4SEEd* werden existierende Erfahrungen gesammelt und neue Ideen entwickelt, um das Potential digitaler Lernspiele in dieser Domäne zu erweitern und zu einer weiter verbreiteten Nutzung in der SE-Ausbildung anzuregen.

Die vorgestellten Ideen dienen als Kernelemente für die Entwicklung einer neuen Spielumgebung, welche das Erlernen von Softwareprozessen effektiv und effizient unterstützt.

Einleitung

Die Entwicklung komplexer Softwaresysteme verlangt nach gut ausgebildeten Softwareingenieuren, welche in der Lage sind, die richtigen Technologien, Werkzeuge und Prozesse auszuwählen, um dynamischen Anforderungen gerecht zu werden.

Zu den großen Herausforderungen heute und in Zukunft gehören dabei die zunehmende Vielfalt und die Forderung nach kürzeren Entwicklungszeiten bei gleichzeitiger Sicherstellung vertrauenswürdiger Qualität (Sommerville, 2010). Die zunehmende Vielfalt umfasst neben Technologien, Plattformen, Werkzeugen und Anwendungsdomänen auch Methoden und Softwareprozesse. Variierende Szenarien erfordern die Berücksichtigung verschiedener Aspekte und das Setzen unterschiedlicher Prioritäten.

Die Ausbildung im Software Engineering (SE) muss dieser Vielfalt Rechnung tragen, um die Softwareingenieure von morgen zu befähigen. Aus Zeitgründen können die Lerninhalte stets nur eine Auswahl aus einem breiten Spektrum sein. Während die Auswahl der Inhalte im Detail variieren wird, ist es breiter Konsens, dass neben allen technologischen Aspekten und Werkzeugen ein fundiertes Wissen über Softwareprozesse essentiell für die erfolgreiche Gestaltung von Softwareprojekten ist.

Motivation

Aufkommende und sich in den Vordergrund drängende agile Prozesse propagieren den Verzicht auf Ballast – die Nicht-Nutzung von Methoden und Werkzeugen, welche wichtige Bausteine traditionellerer Entwicklungsprozesse bilden. Ohne Erfahrung in der Entwicklung komplexerer Softwaresysteme können die Konsequenzen der Nicht-Nutzung von Methoden und Werkzeugen jedoch nur schwerlich qualifiziert eingeschätzt werden. Das Kennenlernen verschiedener Softwareprozesse kann an dieser Stelle helfen, Methoden und Werkzeuge sowie deren Wirkung besser einschätzen zu können.

Durch Lehrende gesteuerte Kursprojekte ergänzen heute gewöhnlich klassische Vorlesungen im Bereich des SE. Akademische Rahmenbedingungen, wie begrenzte Zeit und die Notwendigkeit, individuelle Leistungen der Studierenden zu bewerten, begrenzen Größe, Umfang und Typen solcher Projekte. Um eine positive Lernerfahrung der Studierenden und einen positiven Projektabschluss zu ermöglichen, wird durch Lehrende häufig eine Vorauswahl von geeigneter Technologie, Werkzeugen, Methoden und Prozessen getroffen, der die Studierenden dann folgen sollen. Die Erhö-

hung des Projektumfangs und der Anzahl der Projektmitglieder, die Anhebung des Kommunikationsbedarfs, eine stärkere Teilung von Aufgaben und Verantwortlichkeiten sowie das Beharren auf einem definierten Projektergebnis erhöhen die Realitätsnähe und vermitteln Studierenden idealerweise einen Eindruck von echter Projektarbeit nach dem Studium.

Der Blick auf den Softwareprozess als Ganzes geht in solchen Projekten jedoch allzu schnell verloren, wenn Projektmitglieder unter Zeitdruck damit beschäftigt sind, Projektartefakte zu liefern. Studierende erhalten nicht die Möglichkeit in verschiedene Rollen zu schlüpfen, um bspw. in der Rolle eines Projektmanagers Zielkonflikte zu spüren und richtungweisende Entscheidungen treffen zu müssen. Die begrenzte zur Verfügung stehende Zeit ermöglicht es nicht, alternative Strategien zu verfolgen oder gar andere Softwareprozesse auszuprobieren.

Sommerville betont, „engineering is all about selecting the most appropriate method for a set of circumstances“ (Sommerville, 2010, S. 7). Um dies zu ermöglichen, müssen zukünftige Softwareingenieure ihre Optionen kennen. Sie müssen in der Lage sein, vielfältige Aspekte und Perspektiven zu berücksichtigen. Simulation und Digital Game-Based Learning (DGBL) können neben Kursprojekten einen wichtigen Beitrag dabei leisten, Softwareprozesse kennenzulernen.

Simulation und digitale Spiele in der Software Engineering Ausbildung

Ziel der SE-Ausbildung sollte es sein, ein tiefes Verständnis von Inhalten und deren Aufnahme in den eigenen Wertekanon zu fördern. (Ludewig, 2009). Folgende Aussage zeigt, wie dies gelingen kann: „Jeder Sinn, den ich selbst für mich einsehe, jede Regel, die ich aus Einsicht selbst aufgestellt habe, treibt mich mehr an, überzeugt mich stärker und motiviert mich höher, als von außen gesetzter Sinn, den ich nicht oder kaum durchschaue…“ (Reich, 2008, S. 95). Folgt man diesem idealtypischen Grundsatz konstruktivistischer Didaktik, so besteht die Aufgabe einer geeigneten Lernumgebung darin, die Eigenaktivitäten der Lernenden anzuregen, um die aktive Konstruktion von Wissen bei der Bearbeitung komplexer authentischer Situationen und Probleme zu unterstützen. Die Betrachtung des Lerngegenstands aus verschiedenen Perspektiven, Artikulation und Reflexion im sozialen Austausch tragen grundlegend zum erfolgreichen Lernen bei (Kritzenberger, 2005). Verschiedene Ansätze aus dem konstruktivistischen Methodenbaukasten finden in der SE-Ausbildung bereits Anwendung (Hagel, Mottok, 2011). Simulationen und digitale Spiele bieten in diesem Umfeld besonders geeignete Merkmale.

Simulation eröffnet Wege, relevante Aspekte einer (fast) realen Welt mit den flexiblen Fähigkeiten von Simulationsexperimenten zu verbinden. Zu diesen Fähigkeiten gehört die Möglichkeit,

- Zeit zu komprimieren oder auszudehnen,
- Quellen von Variationen gezielt zu steuern,
- Experimente zu beliebigen Zeitpunkten anzuhalten und zu analysieren,
- Systemzustände (wieder-)herzustellen um Experimente zu wiederholen, sowie
- den Detaillierungsgrad zu definieren.

Ergänzt man Kursprojekte um Lernformen mit diesen Merkmalen, so lässt sich ein hohes Maß an Flexibilität gewinnen. Studierende und Lehrende sind in diesem Fall nicht an das operative Erfordernis gebunden, tatsächlich Software zu produzieren, um den damit verbundenen Prozess lebhaft zu erfahren. Die gezielte Erkundung verschiedener Softwareprozesse in verschiedenen Szenarien wird damit auch in einem begrenzten Zeitfenster möglich. Da keine Ergebnisse in einem echten Kursprojekt gefährdet werden, kann ungezwungenes spielerisches Experimentieren Teil der Lernerfahrung werden. Studierende können auch in (simulierten) Situationen agieren, welche aufgrund begrenzter Ressourcen real nicht denkbar wären.

“Simulations are most engaging when made into games” (Prensky, 2007, S. 225). Spiel und Nachahmung bleiben als natürliche Lernstrategien lebenslang wichtige Akkommodations- und Assimilationsstrategien (Rieber, 1996). Der fesselnde und motivierende Charakter von Spielen fördert die intrinsische Motivation der Lernenden, indem er sie dazu motiviert, die Verantwortung für den eigenen Lernfortschritt zu übernehmen (Akilli, 2007). Jede Verbindung von Ausbildungsinhalt *(educational content)* und digitalen Spielen wird dabei als *Digital Game-Based Learning (DGBL)* bezeichnet (Prensky, 2007). DGBL fördert den Lernprozess, indem durch die Integration attraktiver Spielelemente – wie Interaktivität, Herausforderungen, kontinuierlichem Feedback, Multimedialität, dem Gefühl der Selbstwirksamkeit, Wettbewerb und Belohnungen – eine motivierende und fesselnde Lernumgebung bereitgestellt wird. „Digitale Spiele können demnach […] ein selbstgeleitetes Lernen durch Exploration ermöglichen und befördern.“ (Breuer, 2010, S. 12)

Bisherige Ansätze

Wir finden heute bereits eine Reihe von verfügbaren Anwendungen von Simulation und DGBL für den Bereich des SE. Tabelle 1 fasst Charakteristik und Fähigkeiten verschiedener Ansätze zusammen, in welchen es explizit um SE-Inhalte, Softwarepro-

zesse und deren Elemente geht. Nicht alle in der Tabelle aufgeführten Projekte sind für den direkten Einsatz in der SE-Ausbildung verfügbar. Einige Ansätze werden nicht mehr aktiv entwickelt. Die in den jeweiligen Projekten gewonnenen Erkenntnisse sind jedoch auch in diesen Fällen in Publikationen dokumentiert. In der letzten Spalte der Tabelle werden den bestehenden Ansätzen die Ziele des Projekts *Sim4SEEd* gegenübergestellt.

	SimSE (Navarro, 2006)	SimjavaSP (Shaw, Dermoudy, 2005)	SESAM (Drappa, Ludewig, 2000) /AMEISE (Mittermeir u. a., 2003)	Incredible Manager (Barros u. a., 2006)	OSS (Sharp, Hall, 2000)	MO-SEProcess (En Ye u. a., 2007)	SimVBSE (Jain, Boehm, 2006)	Sim4SEEd* (Pieper, 2012)
Einzel(E)/Mehrspieler(M)-Spiel	E	E	E	E	E	M	E	M
Zusammenarbeit und Wettbewerb unter Spielern	-	-	-	-	-	o	-	+
Dashboards für Team, Kurs und Lehrende	-	-	-	-	o	-	-	+
Spieler in der Rolle eines Projektmanagers	+	+	+	+	-	-	+	+
Spieler mit verschiedenen Aufgaben	-	-	-	-	+	+	-	o
Wiederholbarkeit von Spielabschnitten	+	-	+	-	-	-	-	+
Ermöglicht Abbildung verschiedener Softwareprozesse	+	-	+	-	-	-	-	+
Anzahl verfügbarer Softwareprozesse	6	1	1	1	-	1	1	
Domain Specific Language (DSL) für die Prozessmodellierung	-	-	+	-	-	-	-	o
Regeleditor für die Prozessmodellierung	+	-	-	-	-	-	-	o
Trennung von Softwareprozess und Anwendungsszenarios	-	-	-	-	-	-	-	+
Integration von Industriestandards für die Beschreibung des Softwareprozesses	-	-	-	-	-	-	-	+
Grafische Repräsentation der Nicht-Spieler-Figuren mit welchen interagiert wird	+	o	-	o	+	+	o	+
Nutzung echter Werkzeuge aus der Softwareentwicklung	-	-	-	-	o	-	-	o
In Komponenten für teilweise Wiederverwendung entworfen	-	-	-	-	-	-	-	+
Öffentlich verfügbar	+	-	+	+	-	-	+	+

"+" = ja/enthalten, "-" = nein/nicht enthalten, "o" = teilweise enthalten, "*" = Projektziele

Tabelle 1: Vergleich von DGBL-Anwendungen

Die typische Aufgabe für einen Spieler besteht in diesen Lernspielen i.d.R. darin, als Projektmanager simulierte Softwareentwickler erfolgreich durch ein simuliertes Softwareprojekt zu dirigieren. Dazu werden Personen in das Projektteam aufgenommen oder aus diesem entfernt. Den simulierten Figuren im Team werden dann verschiedene Aufgaben, welche mit dem abgebildeten Softwareprozess korrespondieren, zugewiesen. Dabei entscheidet der Spieler, wer wann welche Aufgabe im Softwareprozess erhält. Die Aufgaben umfassen SE-Tätigkeiten, die Auswirkungen auf den Fortschritt und die Qualität eines Softwareprojekts haben. Das eingesetzte virtuelle Personal wird virtuell entlohnt. Erfolgreich ist, wer sein virtuelles Projekt fristgemäß und ohne Budgetüberschreitung in akzeptabler Qualität abliefert.

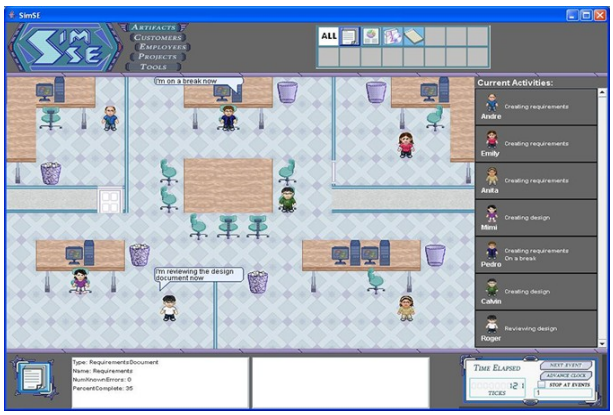

Abb. 1: *SimSE* im Einsatz

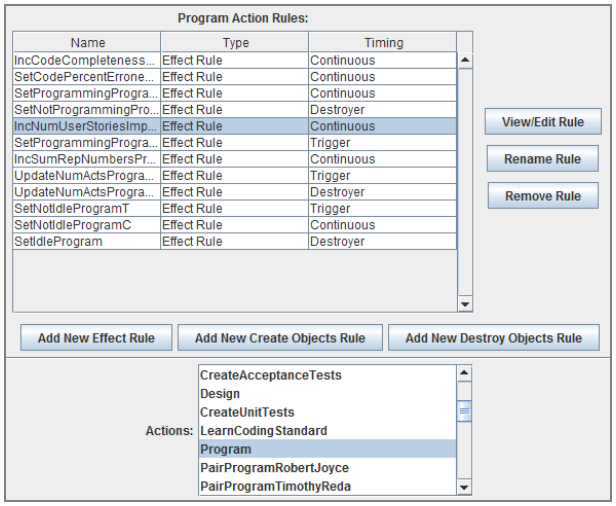

Abb. 2: Ausschnitt aus *SimSE's* Regeleditor

Abbildung 1 zeigt einen Screenshot von *SimSE* (Navarro, 2006), einer an der University of California, Irvine, entwickelten Spielumgebung, welche aktuell die Erkundung der zahlreichsten Softwareprozesse erlaubt. Die simulierten Soft-

wareentwickler mit ihren verschiedenen Eigenschaften, welche durch das simulierte Softwareprojekt gesteuert werden, sind neben bereits erstellten Artefakten zu erkennen. Abbildung 2 stellt den grafischen Regeleditor von *SimSE* dar, über welchen sowohl der Softwareprozess als auch das spezifische Anwendungsszenario modelliert werden. Sichtbar ist hier nur ein kleiner Ausschnitt der modellierten Regeln des Simulationsmodells, welches die Basis des Spiels darstellt.

Ergebnisse bisheriger Forschungsprojekte

Die Auswertung bisheriger Forschungsprojekte zeigt großes Potential aber auch einige Ambivalenzen. Bisherige Ansätze scheinen besser geeignet, bereits erworbenes Wissen zu festigen, als neues Wissen zu vermitteln (Wangenheim, Shull, 2009). Einige Projekte waren nicht in der Lage eine Wirkung von spielbasiertem Lernen nachzuweisen. Andere ermutigende Studien zeigten, dass der Einsatz von Lernspielen geeignet war, auf Seiten der Studierenden Empathie für die Notwendigkeit von Methoden und Prozessen im SE zu entwickeln. Die Lernenden fanden eingesetzte Spiele fesselnd, einfach zu benutzen und zogen sie traditionellen Lehrmethoden vor. Die sorgfältige Planung und Einbettung von Spielaktivitäten in das Curriculum, ausreichende Anleitung, detailliertes Feedback, Diskussion sowie die gründliche Auswertung und Erklärung der Spielresultate werden dabei als essentiell wichtig erachtet (Wangenheim, Shull, 2009).

Die verwendete Methodik der Evaluierungen dieser spielbasierten Ansätze wird durchaus kritisch bewertet (Wangenheim, Shull, 2009), was auch in anderen Anwendungsdomänen von spielbasiertem Lernen der Fall ist (Rieber, 1996; S. Egenfeldt-Nielsen, 2007).

Das Potential ist bei weitem noch nicht ausgeschöpft.

Die meisten verfügbaren digitalen Lernspiele sind ausschließlich für Einzelspieler konzipiert. Als solche fordern Sie von jedem Spieler die Auseinandersetzung mit dem gesamten abgebildeten Softwareprozess, fördern also den Blick auf den Prozess als Ganzes. Jedoch mangelt es bei einer isolierten Spielerfahrung an motivierender Zusammenarbeit und an motivierendem Wettbewerb.

Die Architekturen der Spiele erlauben es Lehrenden nicht, sich einen schnellen Überblick über den bisherigen Fortschritt aller Spieler im Kurs zu erhalten.

Die meisten verfügbaren Ansätze unterstützen nur einen spezifischen Softwareprozess. Angesichts aktueller Entwicklungen und der Vielfalt verfügbarer Prozesse erscheinen die Anpassungsfähigkeit und die Auswahlmöglichkeit verschiedener Prozessmodelle äußerst wichtig.

Die Erzeugung und Anpassung von Simulationsmodellen erfordern einen erheblichen Einarbeitungsaufwand, der außerhalb der jeweiligen Spielwelt kaum direkt von Nutzen ist.

Es mangelt an einer Trennung von Softwareprozessmodell und dem konkreten Anwendungsszenario dieses Prozesses im Spiel, was eine Wiederverwendung erschwert. Visualisierungen der Softwareprozesse werden nicht angeboten. Diese würden die Validierung der schnell komplex werdenden Modelle enorm erleichtern und Transparenz erzeugen. Lehrende könnten leichter überblicken, ob das Modell den Softwareprozess korrekt abbildet und die gewünschten Lerninhalte transportiert. Visualisierung in Kombination mit der Trennung von Softwareprozess und seinem Anwendungsszenario im Spiel wäre als unterstützendes Lernmaterial auch für die Studierenden hilfreich. Die Einbindung von unterstützendem Lernmaterial über den jeweiligen Softwareprozess in die Spielwelt wird in existierenden Ansätzen nicht unterstützt.

Sim4SEEd – Kernelemente einer neuen digitalen Spielumgebung für Softwareprozesse

Nach einer Analyse existierender Ansätze und Anwendungen wurde eine Reihe von Kernelementen identifiziert, welche die Basis einer neuen webbasierten Spielumgebung bilden. Diese wird aktuell im Rahmen des Forschungsprojekts *Sim4SEEd – **Sim**ulation and Digital Game-Based Learning for **S**oftware **E**ngineering (Process) **Ed**ucation* entwickelt. Dabei werden zwei wesentliche Ziele verfolgt:

1. Um Lernerfolg zu maximieren, wird die Spielumgebung die Anforderungen an konstruktivistische Lernumgebungen umfassender als bisherige Ansätze unterstützen. Die Unterstützung sozialer Interaktion, frühes Feedback als Anreiz für Artikulation und Reflexion sowie die Bereitstellung multipler Perspektiven auf den jeweiligen Softwareprozess bilden hierbei Schwerpunkte.
2. Auch gute Lösungen finden nur dann ihren Weg in die Praxis, wenn sich ihre Nutzung für Anwender nicht unnötig kompliziert gestaltet. Lehrende werden unter Ausnutzung von Synergien von der transparenten Erstellung des Simulationsmodells über die Administration des Lernspiels bis hin zu kursweiten Auswertungen unterstützt.

Die nun folgenden Abschnitte beschreiben die Kernelemente, welche zur Erreichung dieser Ziele beitragen werden.

Soziale Interaktion als Kollaboration und Wettbewerb

Die Kombination aus Einzelspiel, in dem alle Entscheidungen selbst getroffen werden, und motivierenden sozialen Elementen ist wünschenswert. Das Hinzufügen eines Teams zum individuellen Spiel jedes Einzelnen erscheint hier erfolgsversprechend. Dabei erkunden und bearbeiten alle Spieler individuell einen gesamten Softwareprozess. Gleichzeitig jedoch agiert jeder Spieler auch in einem Team und sammelt für dieses Punkte. Indem die Performance des Teams – die Kumulation der Einzelspielerergebnisse – als primärer Erfolgsfaktor herangezogen wird, werden Zusammenarbeit und Diskussion innerhalb der Teams gefördert.

Dashboards, welche einerseits Teamrankings und andererseits Rankings einzelner Spieler innerhalb dieser Teams zur Verfügung stellen, bieten dabei Orientierung. Sie zeigen an, wie gut bisher getroffene Entscheidungen in Relation zu anderen Spielern waren. Diese Orientierung dient als Ausgangspunkt für Interaktionen innerhalb der Teams und als Anlass für den ständigen Versuch, Entscheidungen zu optimieren.

nicht geboten werden kann. Lernende erhalten Feedback zu einem viel früheren Zeitpunkt, als dies in einem realen (Kurs-)Projekt der Fall wäre.

Mehr noch – durch die vorgeschlagene Kombination aus Einzelspiel und Zusammenarbeit im Team, kann parallel auch aus Erfahrungen anderer Spieler gelernt und profitiert werden. Das Spiel bietet damit die Basis für einen *„probe, hypothesize, reprobe, rethink cycle“* (Gee, 2007, S. 87 ff.), der zu einer tieferen Auseinandersetzung mit dem simulierten Softwareprozess führt. Die im Spiel gebotene Transparenz geht über die in vielen Projekten vorgefundene Realität hinaus und ist im Idealfall ein Anstoß für das Hinterfragen von Intransparenz in erlebten oder noch folgenden (Kurs-)Projekten der Lernenden.

Abbildung 3 zeigt einen Mockup-Entwurf des webbasierten Spiels mit Kollaborationselementen. Im Kopf der Seite ist erkennbar, dass der Spieler Austin Mitglied des Teams 3 ist. Auf der rechten Seite befinden sich Informationen über das aktuelle Ranking, ein Team-Chat und Nachrichten vom Spielleiter. Das Ranking stellt dar, dass sich das Team 3 momentan an Platz 3 (von 10 Teams) befin-

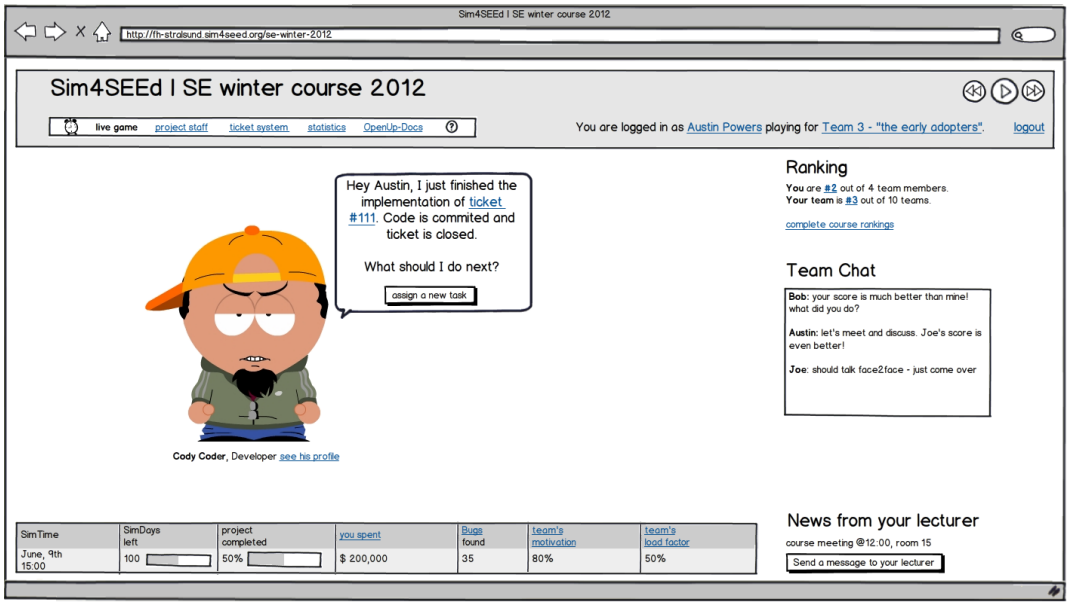

Abb. 3: Mockup mit Kollaborationselementen

Dazu bietet das Spiel die Möglichkeit, einzelne Spielabschnitte zu wiederholen. Wiederholbarkeit in Kombination mit verfügbarer Orientierung unterstützt die Analyse und Reflexion getätigter Schritte und fördert somit die Anwendung des im Spiel erworbenen Wissens. Gleichzeitig stärkt Sie das Gefühl der Selbststeuerung *(control)* der Spielenden und steigert damit deren Motivation.

An dieser Stelle zeigt sich eine Qualität, welche in Kursprojekten in dieser komprimierten Form

det. Auf das Team bezogen drückt es aus, dass Austin innerhalb seines Teams von vier Teammitgliedern am zweitbesten agiert hat. Ein Teammitglied hat bisher also noch bessere Entscheidungen getroffen. Dies ist in diesem Beispiel Anlass für eine kleine Diskussion im Team-Chat. Die Teammitglieder verabreden sich zu einem persönlichen Treffen, in welchem Entscheidungen analysiert und optimiert werden.

Eine Architektur, welche Lehrenden einen schnellen Überblick über den Spielfortschritt des gesamten Kurses bietet, unterstützt die gezielte Interaktion bei Verständnisproblemen sowie die kursweite vergleichende Auswertung und reflektierende Nachbesprechung der Spielergebnisse.

Nutzung von Synergien mit Industriestandards

Der einheitliche Zugang zu Softwareprozessen wird durch deren verschiedene Darstellungsformen erschwert. Die Softwareindustrie hat einigen Aufwand in die vereinheitlichte Dokumentation und Kommunikation von Softwareprozessen investiert. Basierend auf der *Software and Systems Process Engineering Metamodel Specification (SPEM) Version 2.0* (Object Management Group, 2008) und ihrer Vorgängerin *Unified Method Architecture (UMA)* stehen mit dem *Eclipse Process Framework (EPF)* (The Eclipse Foundation, 2012) Werkzeuge bereit, um Softwareprozesse und Methoden zu dokumentieren, zu konfigurieren und zu veröffentlichen.

Inhalte werden mit den EPF-Werkzeugen standardisiert und relativ komfortabel erstellt. Dabei kommt ein gemeinsames einheitliches Vokabular mit definierter Semantik zum Einsatz. Der in der Abbildung 4 dargestellte *EPF Composer* verwendet die vertraute *Unified Modeling Language (UML)* um grafische Visualisierungen der modellierten Softwareprozesse zu erzeugen. Diese Visualisierungen unterstützen die komfortable Erkundung der erzeugten Prozessmodelle.

Der hierbei erbrachte Einarbeitungsaufwand der Lehrenden und Lernenden ist über eine spezifische Simulations- und Spielwelt hinaus von Wert und Nutzen.

Unterstützung der Erkundung neuer Softwareprozesse

Die Evaluation bestehender Ansätze digitaler SE-Lernspiele hat ergeben, dass diese weniger geeignet waren, neue Lerninhalte zu vermitteln (Wangenheim, Shull, 2009). Ein Grund dafür dürfte sein, dass diese Umgebungen die Einbettung von Lern-

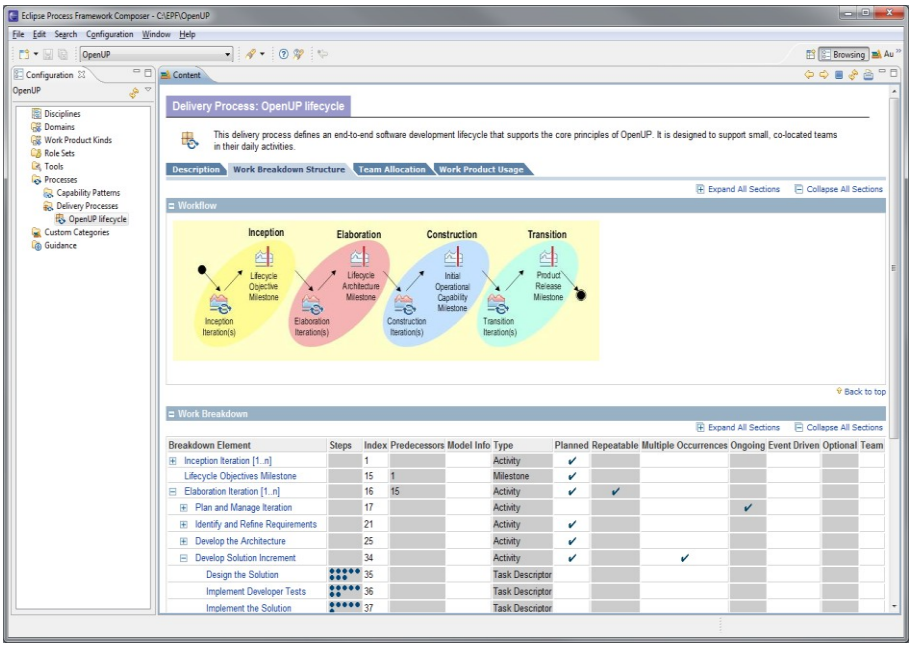

Abb. 4: EPF Composer im Einsatz

EPF stellt über diese Werkzeuge hinaus bereits eine umfangreiche Bibliothek von Softwareprozessinhalten bereit.

Die Nutzung dieser Industriestandards zur Generierung (von Teilen) der Simulationsmodelle als Basis digitaler Spiele bietet eine Reihe von Vorteilen. Rollen, Phasen, Aufgaben, Prozessschritte, Artefakte, Templates und Leitfäden können extrahiert und in die Simulations- und Spielwelt eingebettet werden. Bereits vorhandene Inhalte der EPF Bibliothek werden dabei effizient genutzt. Neue material nicht unterstützen.

Die Verwendung des *Eclipse Process Framework (EPF)* bietet sich auch an, um Lerninhalte eines Softwareprozesses in die Simulations- und Spielwelt zu integrieren. So bietet der *EPF Composer* die Möglichkeit, den dokumentierten Prozess in ein HTML-Format zu exportieren, welches dann, lokal oder auf einem Webserver veröffentlicht, ohne weitere Werkzeuge mit einem Webbrowser erkundet werden kann.

Nach vorheriger Aufbereitung der exportierten Softwareprozessbeschreibung ist es somit möglich, diese für die Erkundung im Stile eines *Adventures* zu verwenden. *Adventures* sind ein Spiele-Genre, in welchem Spieler aktiv ihre Spielwelt erkunden und dabei typischerweise Gegenstände und Informationen sammeln, welche sie im weiteren Spielverlauf anwenden und kombinieren, um gestellte Rätsel und Aufgaben zu lösen. Übertragen auf ein digitales Lernspiel in der SE-Ausbildung bedeutet dies, dass ein Spieler die Prozessbeschreibung erkundet und dabei Informationen und Werkzeuge sammelt, welche anschließend im Spiel verwendet werden können. Eine bestimmte Aufgabe mit einem spezifischen Werkzeug kann erst dann an eine simulierte Software-Entwicklerin übertragen werden, wenn zuvor die zugehörige Dokumentation der Aufgabe in der Prozessbeschreibung besucht und das entsprechende Werkzeug „eingesammelt" wurde.

Da die Spielumgebung die Erkundungsaktivitäten der Spieler verfolgt, erhalten auch Lehrende einen Überblick darüber, wie aktiv Einzelspieler, Teams und der gesamte Kurs im Spiel voranschreiten. So werden Verständnisprobleme frühzeitig erkannt und durch Interaktionen zwischen Lehrenden und Lernenden aufgelöst.

Durch die standardisierte Struktur der Prozessbeschreibungen fällt Studierenden die Erkundung eines zweiten oder dritten Softwareprozesses zunehmend leichter. Sie erhalten spielerisch Zugang zum Kennenlernen verschiedener Softwareprozesse in einem Industriestandard.

Effiziente Erzeugung transparenter Simulationsmodelle

Bisherige Ansätze verwenden eine *Domain Specific Language (DSL)* (Drappa, Ludewig, 2000) oder einen Regeleditor mit grafischer Oberfläche (Navarro, 2006). Die Entwicklung der Modelle erfolgt in den verfügbaren Spielumgebungen von Null an, d.h. jedes einzelne Attribut einer jeden Entität und jedes Element des Softwareprozesses muss aufs Neue modelliert werden. Auch die Steuerung der zugrundeliegenden Simulation muss über Regeln implementiert werden. Ein Konzept der Wiederverwendung existiert hierbei nicht. Erschwerend kommt hinzu, dass in den Modellen nicht zwischen dem eigentlichen Softwareprozess und seiner Anwendung in einem spezifischen Kontext (Spielszenario, Simulationsexperiment) unterschieden wird.

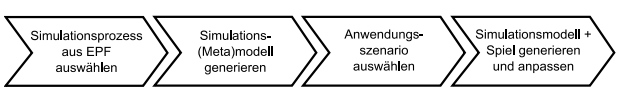

Abb. 5: Erzeugung eines Simulationsmodells

Abbildung 5 stellt einen anderen Entwurfsprozess für derartige Simulationsmodelle dar. In diesem mehrstufigen Modellierungsprozess werden in einem ersten Schritt Inhalte aus der EPF Bibliothek ausgewählt bzw. mit dem *EPF Composer* erzeugt. In einem zweiten Schritt wird daraus ein noch unvollständiges Simulations(meta)modell generiert. Dazu werden Rollen, Aufgaben, Prozessschritte, Artefakte, Leitfäden etc. aus dem EPF-Modell extrahiert. In einem dritten Schritt wird ein passendes Szenario für die Anwendung dieses Softwareprozesses ausgewählt. Dieses Szenario charakterisiert das simulierte Projektumfeld und enthält bspw. Details über verfügbare simulierte Softwareentwickler. Im vierten Schritt wird aus dem gewählten Szenario und dem teilfertigen Simulations(meta)modell ein Simulationsmodell und ein darauf basierendes Spiel generiert, welches anschließend für den konkreten Einsatz konfiguriert und angepasst werden kann. An dieser Stelle werden Parameter eingestellt oder Exkurse in Form von Minispielen in das Spiel integriert. Ziel dieses vorgestellten Entwurfsprozesses ist es, den Aufwand des Lehrenden für die Erstellung eines Simulationsmodells gegenüber bestehenden Ansätzen zu verringern. Die Tätigkeit des Modellerstellers wird sich dabei idealerweise auf das Mapping des gewählten Softwareprozessmodells auf das Anwendungsszenario und die Parametrisierung der Simulation bzw. des Spiels beschränken.

Nutzung von Werkzeugen aus dem echten Softwareentwicklerleben

Aktuelle kommerzielle Spieletitel begeistern Spieler mit fotorealistischen Darstellungen virtueller Welten. Einen Wettbewerb um Realitätsnähe auf dieser Ebene kann ein digitales Lernspiel mit viel geringerem Budget und Ressourceneinsatz nur verlieren. Wie kann ein Lernspiel jenseits solcher Dimensionen dennoch relevant und realistisch wirken?

Die Integration von Werkzeugen aus dem echten Softwareentwicklerleben in das Spielerlebnis trägt an dieser Stelle dazu bei, den realistischen Charakter des Lernspiels zu unterstreichen. Diese Werkzeuge werden durch simulierte Softwareentwickler getrieben. Hierzu ein kleines Beispiel: der simulierte Softwareentwickler Bob hat gerade seine Arbeit am Anforderungsdokument beendet. Er stößt ein Commit in der verwendeten Versionsverwaltung an und schließt anschließend das Ticket im Ticket-Verwaltungssystem – so wie in einem echten Softwareprojekt. Der Spieler sieht diese Aktivitäten anschließend in den Aktivitäten- und Repository-Sichten der jeweiligen Werkzeuge und kann sie nachvollziehen. Aktuelle Werkzeuge, wie bspw. das an der FH Stralsund eingesetzte *Redmine* (Lang, 2012), sind quelloffene Webanwendungen und verfügen über Schnittstellen für die Integrati-

on, so dass sie auch von simulierten Softwareprojekten verwendet werden können.

Studierende, welche bereits mit derartigen Werkzeugen gearbeitet haben, werden diese wiedererkennen und Parallelen herstellen können. Anderen Studierenden werden sie im Laufe ihres Studiums noch begegnen und dann schon etwas vertrauter vorkommen.

Förderung von Analyse und Reflexion

Neben der aktiven Problemlösung, kontinuierlichem Feedback und sozialer Interaktion in Teams tragen verschiedene Perspektiven zu einer konstruktivistischen Lernumgebung bei.

Das aktive Steuern der simulierten Charaktere in der Spielumgebung stellt eine erste Perspektive dar. Die gesammelten Statistiken während des Spiels liefern eine aggregierte zweite Sicht. Integrierte Werkzeuge aus dem echten Softwareentwicklerleben liefern eine dritte realitätsnahe Perspektive. Die verfügbare Beschreibung des Softwareprozesses, welche während des Spiels erkundet wird, stellt eine weitere Perspektive dar, die gleichzeitig vom gespielten Anwendungsszenario abstrahiert.

Da die Spielumgebung mit dem *Eclipse Process Framework (EPF)* auf einem frei verfügbaren Industriestandard basiert, ist es ohne weiteres möglich, Studierende im Anschluss an ein Spiel in dieses einzuführen. Mit dem EPF Composer können bspw. eigene Prozesse modelliert, analysiert und diskutiert werden, was die Reflexion und die Übertragung des erworbenen Wissens auf andere Szenarien und Kontexte fördert.

Bessere Wiederverwendbarkeit und webbasierte Architektur

Die Trennung von Spiel- und Simulationskomponente eröffnet die Möglichkeit, nur einzelne Bestandteile der Umgebung (wieder) zu verwenden. So können Spiele mit einem anderen Spielfluss oder anderen Benutzeroberflächen die gleiche Simulationskomponente nutzen.

Aktuelle Entwicklungen im Bereich der Webtechnologien *(HTML5, CSS3, WebSocket, Server-Sent Events, etc.)* ermöglichen attraktive Spielerfahrungen im Webbrowser – ganz ohne die Nutzung proprietärer Erweiterungen. Die Nutzung des Webbrowsers, in welchem sich heutige Studierende daheim fühlen, vereinfacht gleichzeitig die nahtlose Integration von webbasierten Werkzeugen aus der echten Softwareentwicklung und die Einbindung von Prozessbeschreibungen, welche über das EPF veröffentlicht wurden.

Fazit und Ausblick

Simulation und Digital Game-Based Learning (DGBL) können bei geeigneter Implementierung den Aufbau einer effizienten konstruktivistischen Lernumgebung unterstützen.

Bestehende Lösungen für die Anwendung von Simulation und Digital Game-Based Learning in der SE-Ausbildung haben Pionierarbeit geleistet und zeigen ermutigende Ergebnisse. Sie schöpfen das vorhandene Potential jedoch nicht aus. Anforderungen an die Gestaltung einer konstruktivistischen Lernumgebung sind in Teilen nicht erfüllt. Dies betrifft insbesondere die Unterstützung sozialer Interaktion, frühe Anreize für Artikulation und Reflexion und die Bereitstellung multipler Perspektiven auf den jeweiligen Softwareprozess als Lerngegenstand.

Im Forschungsprojekt *Sim4SEEd* wurden Ideen entwickelt, mit welchen vorhandenes Potential weiter ausgeschöpft wird. Diese Ideen bilden die Kernelemente einer neuen Spielumgebung, welche im Rahmen des Projekts entwickelt wird.

Sim4SEEd (www.sim4seed.org) ist ein laufendes Forschungsprojekt. Der derzeitige Fokus der Projektaktivitäten liegt auf der Konzeption einer passenden Gesamtarchitektur und der Modellierung der Simulationskomponente, welche die Einbindung von Softwareprozessen aus dem *Eclipse Process Framework (EPF)* unterstützt.

Literatur

Akilli, G. (2007): „Games and Simulations: A New Approach in Education?“. In: *Games and Simulations in Online Learning: Research and Development Frameworks*. Information Science Pub., S. 1–20.

Barros, M.O.; Dantas, A.R.; Veronese, G.O. u. a. (2006): „Model-driven game development: experience and model enhancements in software project management education“. In: *Software Process: Improvement and Practice*. 11 (4), S. 411–421.

Breuer, J. (2010): „Spielend lernen? Eine Bestandsaufnahme zum (Digital) Game-Based Learning“. Landesanstalt für Medien Nordrhein-Westfalen (LfM).

Drappa, A.; Ludewig, J. (2000): „Simulation in software engineering training“. In: *Proceedings of the 22nd international conference on Software engineering*. New York, NY, USA: ACM (ICSE '00), S. 199–208, DOI: 10.1145/337180.337203.

En Ye; Chang Liu; Polack-Wahl, J.A. (2007): „Enhancing software engineering education using teaching aids in 3-D online virtual worlds“. In: *Proceedings of the 37th Annual Frontiers In Education Conference , FIE '07*. IEEE, S. T1E–8–T1E–13, DOI: 10.1109/FIE.2007.4417884.

Gee, J.P. (2007): What Video Games Have to Teach Us About Learning and Literacy. Palgrave Macmillan.

Hagel, G.; Mottok, J.; Ludewig, Jochen; Böttcher, Axel (Hrsg.) (2011): „Planspiel und Briefmethode für die Software Engineering Ausbildung-ein Erfahrungsbericht“. In: *SEUH 2011 Software Engineering im Unterricht der Hochschulen*.

Jain, A.; Boehm, B. (2006): „SimVBSE: Developing a game for value-based software engineering“. In: *Proceedings of the 19th Conference on Software Engineering Education and Training*., S. 103–114.

Kritzenberger, H. (2005): Multimediale und interaktive Lernräume. München :: Oldenbourg,.

Lang, J.-P. (2012): „Overview - Redmine“. Abgerufen am 14.11.2012 von http://www.redmine.org/.

Ludewig, J.; Jaeger, Ulrike; Schneider, Kurt (Hrsg.) (2009): „Erfahrungen bei der Lehre des Software Engineering“. In: *Softwareengineering im Unterricht der Hochschulen: SEUH*. 11 .

Mittermeir, R.T.; Hochmüller, E.; Bollin, A. u. a. (2003): „AMEISE – A Media Education Initiative for Software Engineering Concepts, the Environment and Initial Experiences“. In: *Proceedings of the Interactive Computer aided Learning (ICL) 2003 International Workshop*. Villach, Austria.

Navarro, E. (2006): „SimSE: A Software Engineering Simulation Environment for Software Process Education“. Irvine, CA: University of California.

Object Management Group (2008): „OMG - Software & Systems Process Engineering Meta-Model (SPEM), v2.0“. Abgerufen am 26.10.2011 von http://www.omg.org/cgi-bin/doc?formal/2008-04-01.

Pieper, J. (2012): „Learning software engineering processes through playing games“. In: *2012 2nd International Workshop on Games and Software Engineering (GAS)*. Zurich, Switzerland, S. 1 –4, DOI: 10.1109/GAS.2012.6225921.

Prensky, M. (2007): Digital game-based learning. Paragon House.

Reich, K. (2008): Konstruktivistische Didaktik: Lehr- und Studienbuch mit Methodenpool. Beltz.

Rieber, L.P. (1996): „Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games“. In: *Educational Technology Research and Development*. 44 (2), S. 43–58, DOI: 10.1007/BF02300540.

S. Egenfeldt-Nielsen (2007): „Third generation educational use of computer games“. In: *Journal of Educational Multimedia and Hypermedia*. 16 (3), S. 263–281.

Sharp, H.; Hall, P. (2000): „An interactive multimedia software house simulation for postgraduate software engineers“. In: *Software Engineering, International Conference on*. Los Alamitos, CA, USA: IEEE Computer Society, S. 688, DOI: http://doi.ieeecomputersociety.org/10.1109/ICSE.2000.10053.

Shaw, K.; Dermoudy, J. (2005): „Engendering an empathy for software engineering“. In: *Proceedings of the 7th Australasian conference on Computing education - Volume 42*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc. (ACE '05), S. 135–144.

Sommerville, I. (2010): Software Engineering. 9th revised edition. Addison-Wesley Longman, Amsterdam.

The Eclipse Foundation (2012): „Eclipse Process Framework Project (EPF) - Home“. Abgerufen am 26.10.2011 von http://www.eclipse.org/epf/index.php.

Wangenheim, C.G. von; Shull, F. (2009): „To Game or Not to Game?“. *IEEE Software*. 26 (2), S. 92–94.