Keine Panik! Manifest für Softwareentwicklung in Studierendenprojekten

Eschweiler, Mark
mark.eschweiler@uni-koeln.de
Universität zu Köln, Deutschland

Evers, Anna-Maria
aevers1@smail.uni-koeln.de
Universität zu Köln, Deutschland

Kruhl, Dominik
dkruhl@smail.uni-koeln.de
Universität zu Köln, Deutschland

Reuhl, Elisabeth
ereuhl1@uni-koeln.de
Universität zu Köln, Deutschland

Türko#lu, Enes
enes.tuerkoglu@uni-koeln.de
Universität zu Köln, Deutschland

Die in Forschungsprojekten entwickelten Anwendungen besitzen für die Digital Humanities einen hohen Stellenwert. Sie sind nicht nur Werkzeuge zum Lösen der jeweiligen Fragestellungen, sondern können auch als neue digitale Gestaltungsformen von Theoriebildung begriffen werden (Kleymann 2019). Genauso beinhaltet auch ein DH-Studium die Entwicklung zahlreicher Anwendungen, die dem Erlernen sowohl der programmiertechnischen Konzeption und Umsetzung wie auch der Organisation eines Softwareprojekts dienen.

Softwareentwicklung als integraler Bestandteil der Forschungsaktivitäten der Digital Humanities braucht ein geeignetes Vorgehen zur Organisation und Planung (Druskat et al. 2018). Agile Methoden erfahren nicht nur innerhalb der IT-Branche große Aufmerksamkeit, sondern auch im DH-Bereich (Heyer et al. 2019: 177). Eine Anwendung, die im Rahmen von Forschung oder Lehre entwickelt wird, besitzt jedoch andere Anforderungen als in der freien Wirtschaft. Die Umsetzung von Grundsätzen agiler Softwareentwicklung in Forschung und Lehre weist diverse Problemfelder auf. Scrum etwa wurde im nordamerikanischen Raum und für die freie Marktwirtschaft entwickelt, und orientiert sich mit seinen Prinzipien an der „lean production“, die wiederum sehr erfolgreich in japanischen Unternehmen etabliert wurde (Nonaka und Takeuchi 2008: 215-216). Daraus resultierende kulturelle und situative Barrieren und unterschiedliche Arbeitsrechte erschweren eine adäquate Adaption solch einer Methodologie in den europäischen Raum, und im universitären Kontext kommen weitere Problematiken hinzu.

Selbst zwischen Forschenden und Studierenden sind die Rahmenbedingungen sehr divergent. In studentischen Projekten sind Erkenntnisgewinn und Lernerfolg, neben einer fachgerechten technischen Umsetzung, entscheidend für die abschließende Bewertung und den Erfolg eines Projektes. Der Aufwand wird nicht monetär vergütet und auch ein Scheitern mit Erkenntnisgewinn kann ein Erfolg sein. Ein experimentelles Lehrveranstaltungsformat am Institut für Digital Humanities der Universität zu Köln, in dem Masterstudierende als Scrum Master eine Projektgruppe aus Bachelorstudierenden betreuen, stellte diese Schwierigkeiten der Umsetzung agiler Methodologien im Studium dar.

Welche Aspekte aus den agilen Methodologien sowie Manifesten übertragbar sind und wo neue Wege sinnvoller sind, wurde im Rahmen eines Seminars im Sommersemester 2019 von Masterstudierenden – unterstützt von Prof. Dr. Øyvind Eide – diskutiert und evaluiert. Orientierungspunkte boten die Erarbeitung von Lektüre zur agilen Entwicklung, die Reflexion eigener Vorerfahrungen, sowie Beobachtungen und Befragungen der Bachelorstudierenden-Projekte aus der übergreifenden Lehrveranstaltung. Um die Überlegungen festzuhalten und für nachfolgende Studierende nutzbar zu machen, fiel die Wahl, angelehnt an u.a. das „Manifesto for Agile Software Development“ (Beck et al. 2001), auf das Format eines Manifestes. Als „Manifest für Softwareentwicklung in Studierendenprojekten“ wurden schließlich die formulierten Rahmenbedingungen für die Planung und Umsetzung von Softwareentwicklung innerhalb von Gruppenprojekten im universitären Kontext vorgestellt.

Manifest für Softwareentwicklung in Studierendenprojekten

Softwareprodukt und **Lernprozess** sollten sowohl bei der Entwicklung als auch bei der Dokumentation die gleiche Wichtigkeit erfahren. Der Erfolg studentischer Softwareprojekte ist im Gegensatz zu kommerziellen Softwareprojekten nicht nur von einem funktionierenden Endprodukt abhängig, sondern soll gleichermaßen an in ihrem Umfang angemessenen und **nachhaltigen Lernergebnissen** gemessen werden. Ziel studentischer Softwareprojekte ist demnach ein nachweisbares Ergebnis, welches sich sowohl im Softwareprodukt als auch im dokumentierten Lernprozess zeigt.

Die Organisation studentischer Softwareprojekte erfolgt **selbstorganisiert** durch die Projektgruppe. Die Aufgabenverteilung soll dabei als ein Prozess der **Kompetenzverhandlung** begriffen werden. Verhandelt werden sollen die realistische (Selbst-)Einschätzung der

vorhandenen Kenntnisse der teilnehmenden Studierenden, sowie jene für das Projekt relevante Kompetenzen, deren Aneignung noch angestrebt wird.

Eine zutreffende Einschätzung des Arbeitspensums und adäquate Verteilung korrespondierender Aufgaben werden durch **modulare Mikroaufgaben** gewährleistet. Die Definition spezifischer **Meilensteine** und ihr anschließendes Erreichen im Arbeits- und Lernprozess soll durch Modularität vereinfacht werden. Eine modulare Arbeitsweise erlaubt zudem, die Kompetenzverhandlung einzelner Gruppenmitglieder **flexibel** zu gestalten und ein kontinuierliches Arbeitstempo zu gewährleisten. Voraussetzung für die Definition solcher Mikroaufgaben ist es, Abhängigkeiten zu anderen Aufgaben weitestgehend zu vermeiden.

Agilität stellt einen essentiellen Bestandteil des Projektes dar. Sowohl die Organisation als auch die technische Umsetzung müssen es zulassen, dynamisch auf neue Erkenntnisse oder Hindernisse zu reagieren, ohne dabei das Gesamtziel aus den Augen zu verlieren. Dabei gilt: je modularer die Entwicklung abläuft, desto flexibler kann auf Veränderungen reagiert werden.

Agilität ist insbesondere auf Kompetenzen in der **Gruppenkommunikation** angewiesen, welche sich um Transparenz und Zuverlässigkeit bemühen sollte. Hierzu bedarf es auch einer geeigneten gemeinsamen **Kommunikationsplattform** , welche den Austausch zur Organisation und Umsetzung des Projektes in Form **regelmäßiger Treffen** garantiert. Hier soll jedes Mitglied umstandslos den **Status Quo** des Projektes einsehen können. Außerdem muss hier auf die Möglichkeit geachtet werden, zwischenmenschliche Aspekte effizient behandeln und arrangieren zu können. Falls gewünscht, kann über jene Plattform auch dem Dozierenden ein Zugang zwecks Beurteilung erteilt werden.

Ein **lauffähiger und vorzeigbarer Prototyp** sollte möglichst schnell erstellt und dann **kontinuierlich weiterentwickelt** werden. Dabei ist im Sinne der Agilität darauf zu achten, durch eine modulare Arbeitsweise die Aufwandskurven möglichst flach zu halten. Aktuelle Entwicklungen können stets funktionsfähig in das Produkt aufgenommen werden.

Bibliographie

Beck, Kent / Grenning, James / Martin, Robert C. / Beedle, Mike / Highsmith, Jim / Mellor, Steve / van Bennekum, Arie / Hunt, Andrew / Schwaber, Ken / Cockburn, Alistair / Jeffries, Ron / Sutherland, Jeff / Cunningham, Ward / Kern, Jon / Thomas, Dave / Fowler, Martin / Marick, Brian (2001): „Manifesto for Agile Software Development“. Agile Alliance. http://agilemanifesto.org/ [letzter Zugriff 27. September 2019].

Heyer, Gerhard / Kahmann, Christian / Kantner, Cathleen (2019): „Generic tools and individual research needs in the Digital Humanities – Can agile development help?“, in: Draude, C. / Lange, M. / Sick, B. (eds.): *INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik – Informatik für Gesellschaft (Workshop-Beiträge)*. Bonn: Gesellschaft für Informatik e.V. 175-180.

Kleymann, Rabea (2019): „Prototypen als Proto-Theorie? – Plädoyer einer digitalen Theoriebildung“, in: Sahle, Patrick (ed.): *DHd 2019 Digital Humanities: multimedial & multimodal.* Frankfurt am Main 197-200. http://doi.org/10.5281/zenodo.2596095 [letzter Zugriff 27. September 2019].

Nonaka, Ikujiro / Takeuchi, Hirotaka (2008): „A Theory of the Firm’s Knowledge-Creation Dynamics“, in: Chandler, Alfred / Hagstrom, Peter / Sölvell, Örjan: *The Dynamic Firm. The Role of Technology, Strategy, Organization, and Regions*. Oxford: University Press 214-241.

Schwaber, Ken / Sutherland, Jeff (2017): „Der Scrum Guide™. Der gültige Leitfaden für Scrum: Die Spielregeln.“ https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-German.pdf [letzter Zugriff 27. September 2019].

Druskat, Stephan / Czmiel, Alexander / Schrade, Torsten (2018): „Research Software Engineering und Digital Humanities. Reflexion, Kartierung, Organisation“, DHd 2018, Köln, 27.02.2018. https://dh-rse.github.io/dhd-workshop-2018-presentation/ [letzter Zugriff 27. September 2019].