Zielgerichtete Anpassung von Software nach der Evolution von kontextspezifischem Wissen*

Stefan Gärtner[1], Jens Bürger[2], Kurt Schneider[1], und Jan Jürjens[2,3]

[1]Leibniz Universität Hannover, Software Engineering Group, Germany
{stefan.gaertner, kurt.schneider}@inf.uni-hannover.de
[2]TU Dortmund, Software Engineering, Germany
jens.buerger@cs.tu-dortmund.de, http://jan.jurjens.de
[3]Fraunhofer ISST, Dortmund, Germany

Motivation und Problembeschreibung

In der Softwareentwicklung wird zu einer gegebenen Menge von Anforderungen eine passende Softwarelösung gesucht. Die Anforderungen werden dabei in einem domänenspezifischen Kontext interpretiert. Das kontextspezifische Wissen kann im Laufe der Zeit jedoch evolvieren, wodurch die geschaffene Lösung nicht mehr unbedingt die Anforderungen erfüllt. Die somit „veraltete“ Lösung muss dann angepasst werden. Ein Beispiel mit Bezug zur IT-Sicherheit ist die Anforderung, dass für eine Anwendung sensible Daten sicher verschlüsselt sein sollen. Bis in die Mitte der 90er Jahre galt der *Data Encryption Standard* (DES) als relativ sicher und wäre als mögliche Lösung in Betracht gekommen. Aufgrund von heutiger Rechenleistung kann DES mit einer Schlüssellänge von 56 Bit jedoch relativ einfach mit einem Brute-Force-Angriff gebrochen werden. Heutzutage würde somit nicht mehr DES sondern die Weiterentwicklung *Advanced Encryption Standard* (AES) zur Erfüllung der Anforderung genutzt werden. Obwohl sich die Anforderung als solche nicht geändert hat, muss die vorliegende Lösung, die in den 90er Jahren auf Basis des damaligen Wissens entstanden ist, aufgrund der Evolution des kryptographischen Wissens angepasst werden. Um die Anpassung einer Software nach der Evolution von kontextspezifischem Wissen zielgerichtet zu unterstützen, muss analysiert werden, welche Auswirkungen das geänderte Wissen auf die Anforderungen hat und wie diese mit der Lösung zusammenhängen. Im Rahmen des DFG Priority Programme 1593 untersucht das Projekt *SecVolution* [JS12] dieses Problem in Bezug auf Sicherheitsanforderungen und sicherheitsspezifisches Wissen.

In der wissenschaftlichen Literatur finden sich zur Lösung dieses Problems eine Vielzahl von Ansätzen. Diese basieren zumeist auf der natürlichsprachlichen Dokumentation oder semiformalen Modellierung von Entwurfsentscheidungen mit zugrunde liegenden Annahmen und Begründungen [HP13, JS12, DR13, CMM08, Sch06]. Dabei wird häufig die An-

nahme getroffen, dass das entsprechende Wissen von der jeweiligen Personengruppe im Entwicklungsprozess hinreichend dokumentiert wird, da dies bei einer zukünftigen Weiterentwicklung Vorteile bringt. Dokumentation erzeugt jedoch Aufwand für Personen, die gerade bei langlebiger Software meist selbst nicht mehr davon profitieren. Dies kann negative Auswirkungen sowohl auf die Qualität als auch auf die Vollständigkeit der Dokumentation haben. Doch selbst bei Vorliegen einer ordnungsgemäßen Dokumentation muss diese bei einer späteren Anpassung gelesen und in Zusammenhang mit dem geänderten Wissen gebracht werden. Die aufgeführten Argumente schränken die Anwendung der entwickelten Ansätze im industriellen Umfeld stark ein.

Fragestellungen

Im Rahmen des Workshops sollen die folgenden Fragen zur Dokumentation von kontextspezifischem Wissen mit dem Fachpublikum diskutiert werden:

1. Welche Aspekte müssen mindestens dokumentiert werden, damit langlebige Software zielgerichtet weiterentwickelt werden kann, wenn sich das kontextspezifische Wissen ändert?
2. Was sind die Vor- und Nachteile einer Dokumentation in natürlicher Sprache im Gegensatz zu einer formaleren Notation?
3. Kann man dokumentierende Aktivitäten so gestalten, dass die dokumentierende Person einen unmittelbaren Nutzen daraus hat?

Durch die Diskussion soll die Entwicklung zukünftiger Methoden zur Dokumentation und deren Anwendbarkeit im industriellen Umfeld vorangetrieben werden.

Literatur

[CMM08] John M. Carroll, Raymond McCall und Ivan Mistrik. *Rationale-Based Software Engineering*. Springer, 2008.

[DR13] Zoya Durdik und Ralf Reussner. On the appropriate rationale for using design patterns and pattern documentation. In *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*, Seiten 107–116. ACM, 2013.

[HP13] Tom-Michael Hesse und Barbara Paech. Supporting the collaborative development of requirements and architecture documentation. In *3rd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks)*, Seiten 22–26. IEEE, 2013.

[JS12] Jan Jürjens und Kurt Schneider. On modelling non-functional requirements evolution with UML. *Modelling and Quality in Requirements Engineering (Essays Dedicated to Martin Glinz on the Occasion of His 60th Birthday)*, 2012.

[Sch06] Kurt Schneider. Rationale as a by-product. In *Rationale Management in Software Engineering*, Seiten 91–109. Springer, 2006.