# Kompetenzverteilung in langlebigen Softwareentwicklungsprojekten

Volker Gruhn, Christoph Hannebauer, Sebastian Stünkel

paluno – The Ruhr Institute for Software Technology
Universität Duisburg-Essen
Gerlingstr. 16
45127 Essen
{volker.gruhn/christoph.hannebauer}@paluno.uni-due.de
sebastian.stuenkel@stud.uni-due.de

**Abstract:** In langlebigen Softwareentwicklungsprojekten werden einige Programmteile weiterverwendet, während ihre ursprünglichen Entwickler das Projektteam bereits verlassen haben. Im Laufe der Zeit kann sich die Zusammensetzung des Projektteams daher so verändern, dass die Kenntnisse des Projektteams über bestimmte Programmteile sinken. Schlimmstenfalls entstehen verwaiste Programmteile, die kaum noch angepasst oder ersetzt werden können, weil niemand im Projektteam die Anforderungen oder gar die Funktionsweise dieser Programmteile kennt. Daraus ergeben sich zwei Forschungsprobleme: Wie können verwaiste Programmteile erkannt werden? Wie kann durch Kompetenzstreuung diese Verwaisung verhindert werden?

## 1 Einleitung

Je größer das Entwicklerteam eines Softwareprojekts ist, desto kleiner ist der Anteil der Module, mit denen sich jeder einzelne Entwickler auskennt. Bei großen Entwicklerteams besitzt daher gar kein Entwickler detailliertes Wissen über alle Komponenten. Durch den erforderlichen Entwicklungsaufwand sind solche großen Softwareprojekte auch teuer, so dass sich die resultierende Software möglicherweise erst dann amortisiert, wenn sie über einen ausgedehnten Zeitraum eingesetzt wird. Diese beiden Punkte zusammengenommen können dazu führen, dass bei Veränderungen der Teamzusammensetzung wichtige Kompetenz zu Programmteilen verloren geht. Ein Programmteil, zu dem das Projektteam nur niedrige Kompetenz hat, wird verwaistes Programmteil genannt. Änderungen dieser verwaisten Programmteile sind teurer als Änderungen anderer Programmteile.

Durch diese Verwaisung kann sich die Architektur verschlechtern, weil Änderungen nicht am optimalen, jedoch verwaisten Programmteil vorgenommen werden, sondern als Behelfslösung andere Programmteile angepasst werden. Außerdem können beispielsweise dringliche Anforderungen, etwa weil sie sicherheitskritisch sind, nicht mehr schnell umgesetzt werden, wenn sie Änderungen der verwaisten Programmteile erfordern.

## 2 Offene Fragen

Nicht nur die Höhe, sondern auch die Art der Kompetenz unterscheidet sich zwischen den Projektmitgliedern. Die Kenntnis der Funktionsweise eines bestimmten Programmteils ist ein Beispiel für eine Kompetenzform. Die allgemeine Erfahrung mit der eingesetzten Programmiersprache oder Technologie ist eine andere Kompetenzform.

Eine Kompetenzmessung erkennt verwaiste Programmteile oder Programmteile, deren Verwaisung möglich ist. Die Kompetenzstreuung verhindert präventiv die Verwaisung von Programmteilen.

Die Kompetenzmessung quantifiziert die Kompetenz der Teammitglieder zu den einzelnen Programmteilen. Mit diesem Wissen kann auf mögliche Veränderungen der Teamzusammensetzung und den damit verbundenen Kompetenzverlust reagiert werden. Wie kann man die Kompetenzverteilung im Team messen und welche Auswirkungen haben Änderungen des Teams? Gibt es bereits verwaiste Programmteile, die nicht mehr ohne großen Schulungsaufwand modifiziert werden können? Mit welchen Methoden lassen sich diese identifizieren? Welchen Einfluss hat die Nichtinteraktion mit Programmteilen über die Zeit auf die Programmkompetenz der identifizierten Experten für diesen Bereich? Beispielsweise Fritz et al., Schuler und Zimmermann und Nguyen et al. zeigen Methoden, um Kompetenz zu messen [FMH07, SZ08, NND+12].

Bei einer Kompetenzstreuung wird die Kompetenz zu jedem Programmteil über mehrere Teammitglieder gestreut. Dies verhindert Wissensverluste durch Teamveränderungen. Eine Möglichkeit der Kompetenzstreuung sind Schulungen. Wie können diese Schulungen gesteuert werden? Bei langlebigen Projekten kommt es nicht selten zum Einsatz älterer Technologien für die es im Zweifelsfall nur noch wenige und daher teure Experten auf dem Arbeitsmarkt gibt. Eine weitere Möglichkeit der Kompetenzstreuung ist daher die Ersetzung von Programmteilen, die auf solchen älteren Technologien beruhen. Wie lassen sich die Programmteile einer Software identifizieren, sodass eine gezielte Ersetzung der Technologie vorgenommen werden kann, noch bevor der Kompetenzträger das Team verlässt?

## Literatur


[FMH07] Thomas Fritz, Gail C. Murphy und Emily Hill. Does a programmer's activity indicate knowledge of code? In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ESEC-FSE '07, Seiten 341–350, New York, NY, USA, 2007. ACM.

[NND+12] Tung Thanh Nguyen, T.N. Nguyen, E. Duesterwald, T. Klinger und P. Santhanam. Inferring developer expertise through defect analysis. In *Software Engineering (ICSE), 2012 34th International Conference on*, Seiten 1297 –1300, Juni 2012.

[SZ08] David Schuler und Thomas Zimmermann. Mining usage expertise from version archives. In *Proceedings of the 2008 international working conference on Mining software repositories*, MSR '08, Seiten 121–124, New York, NY, USA, 2008. ACM.