Aktionsforschung in der Software-Visualisierung

Rainer Koschke, *AG Softwaretechnik*, *Universität Bremen*

Dieser Beitrag widmet sich der empirischen Forschung in der Software-Visualisierung (SV). Wir setzen uns kritisch mit den bis dato eingesetzten empirischen Methoden auseinander und argumentieren, dass die Aktionsforschung ein komplementärer Ansatz ist, der gerade für das Design und die Evaluation von Software-Visualisierungen vielversprechend ist.

I. Empirische Forschung in der SV

Merino et al. [1] haben 209 Artikel, die bei der *IEEE Working Conference on Software Visualization (VISSOFT)* oder dem *ACM SOFTVIS Symposium on Software Visualization (SOFTVIS)* als *Full Paper* veröffentlicht wurden, ausgewertet. Von den 181 Artikel, die sich einer Visualisierung im engeren Sinne widmen, haben die Autoren diese hinsichtlich der in den Artikeln beschriebenen Form der Evaluation weiter kategorisiert in empirisch (113), rein theoriegestützt (2) oder gar nicht evaluiert (24). Bei den 113 als empirische Evaluation eingestuften Arbeiten traten die folgenden Formen einer Evaluation auf: anekdotische Evidenz (6), Nutzungsszenarien, in denen eine Visualisierung anhand eines Beispiels demonstriert wird (38), Literaturübersichten (4), Fallstudien (12) und Experimente (53). In allen Fallstudien waren professionelle Entwicklerinnen und Entwickler beteiligt. Bei den Experimenten waren die Teilnehmer hingegen meist Studierende. Der Median der Teilnehmerzahlen dieser Experimente lag bei 13. Die meisten Experimente hatten zwischen eins und fünf Teilnehmende.

Literaturübersichten sind keine direkten Evaluationen von Visualisierungen. Anekdotische Evidenz stellt keine wirklich empirisch fundierte Evaluation dar. Die Demonstration einer Visualisierung in Form eines Nutzungsszenarios mag eine Visualisierung an einem konkreten Beispiel gut illustrieren, jedoch sind es in der Regel die Autoren, die ihre eigene Visualisierung vorführen, und nicht davon unabhängige Entwickler und Entwicklerinnen, die die Visualisierung an ihrem eigenen Code erproben. Dieser Form der Evaluation – wenn man sie überhaupt als solche bezeichnen mag – fehlt es in der Regel auch an Systematik und objektiver Beobachtung.

Kontrollierte Experimente sind im Gegensatz zu allen anderen empirischen Methoden in der Lage, Wirkungszusammenhängen nachzugehen, indem sie bestimme Einflussvariablen fixieren und andere systematisch variieren, um deren Auswirkung zu untersuchen. Wie Merino et al. [1] jedoch festgestellt haben, sind Experimente in der Software-Visualisierung in Bezug auf ihre Teilnehmerzahl eher klein. Meist wird hier bei der Teilnehmerauswahl auch nur Convenience-Sampling angewandt und bei Forschenden an Universitäten führt dies überwiegend dazu, dass auf Studierende zurückgegriffen wird. Der Anspruch an die Standardisierung (Kontrolle von Variablen) führt fast notwendigerweise dazu, dass die Teilnehmenden mit der Software, die visualisiert wird, gar nicht vertraut sind. Die Auseinandersetzung der Teilnehmenden mit der Software und ihrer Visualisierung ist meist einmalig und für sie folgenlos. In der realen Welt haben solche Dinge jedoch potentiell einen Einfluss auf die weitere Entwicklung der Software. Die Teilnehmenden verwenden üblicherweise auch die Visualisierungform zum ersten Mal. Deshalb werden sie im Experiment erst einmal einem Training in der Visualisierung und der damit verbundenen Interaktionen unterzogen. Der Ökonomie geschuldet, ist die Zeit hierfür jedoch meist sehr begrenzt. Die Teilnehmenden sind schon gar nicht beim Design involviert gewesen, so dass ihre individuellen Bedürfnisse in der Visualisierung allenfalls zufällig berücksichtigt sind. Eine weitere Schwäche von Experimenten ist es, dass sie recht punktuelle Untersuchungsformen sind. Nach einem Experiment vergeht eine ganze Zeit bis zum Nachfolgeexperiment, bei dem dann aber die Teilnehmenden in der Regel wieder andere sind. Dieser Untersuchungsform fehlt dann Kontinuität und Langfristigkeit.

Bei Fallstudien beobachten Forschende passiv, wie eine Software-Visualisierung von Entwicklern im eigenen Arbeitskontext verwendet wird. Der Begriff *Fallstudie* drückt es aus: man studiert hier einen Fall in der realen Welt, ohne Kontrolle auszuüben. Insofern haben Fallstudien einen höheren Grad an Realismus als kontrollierte Experimente, die in einer Laborumgebung stattfinden. Während dieses Vorgehen in Kontexten adäquat sein kann, in denen es vornehmlich darum geht, ein Phänomen zu verstehen, geht es jedoch in der Software-Visualisierung in den meisten Fällen gerade darum, eine Visualisierung so zu gestalten, dass sie möglichst nutzbringend ist. Wir wollen als Forschende hier tatsächlich in die reale Welt eingreifen.

II. Aktionsforschung

Im Software-Engineering steckt der Begriff *Engineering*, also das Ziel, ein reales Problem zu lösen, in unserem Falle eines im Zusammenhang der Software-Entwicklung. Würden wir als Forschende im Software-Engineering so vorgehen, wie wir es auch in der Lehre unseren angehenden Software-Engineers vermitteln, dann würden wir zunächst einmal eine genaue Bedarfsanalyse durchführen, um zu ermitteln, wer welche Information zu welchem Zweck benötigt. Im Sinne einer partizipativen Entwicklung würden wir die späteren Nutzer möglichst früh einbinden und unsere Konzepte mittels Prototypen

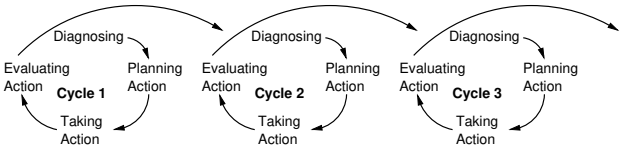

Fig. 1. Vorgehen in der Aktionsforschung

erproben. Als evolutionäre Prototypen würden wir diese iterativ und inkrementell weiterentwickeln, bis sie die Anforderungen erfüllen. Auch danach im Betrieb würden wir weitere Erfahrungen sammeln, die uns helfen, die Software-Visualisierung kontinuierlich weiterzuentwickeln.

Partizipative, agile, iterative und inkrementelle Software-Entwicklung hat viele Parallelen mit der Aktionsforschung. Die Aktionsforschung (engl. *Action Research*) stammt ursprünglich aus den Sozialwissenschaften. Sie wurde für Kontexte eingeführt, in denen die Forschenden nicht nur ein soziales Phänomen beobachten, sondern auch positiv auf es einwirken wollen. Die Aktionsforschung ist mittlerweile aber auch eine anerkannte Methodik in der empirischen Softwaretechnik [2], wenngleich sie in der Darstellung „kanonischer“ Methoden in der empirischen Softwaretechnik gewöhnlich nicht auftaucht.

Die Aktionsforschung übertragen auf die Softwaretechnik verbindet die guten Praktiken des Software-Engineerings mit dem empirischen Erkenntnisgewinn. Im Gegensatz zu Experimenten ist dies eine Untersuchung über einen längeren Zeitraum mit hohem Realismus, bei dem die Teilnehmenden von Anfang an eingebunden sind und die Visualisierung im eigenen Arbeitskontext am eigenen Code evaluieren. Im Gegensatz zu reinen Fallstudien greifen Forschende hier ein, indem sie die Visualisierung kontinuierlich und bedarfsgerecht anpassen. Die Aktionsforschung ist aber keineswegs ein Ersatz für andere Untersuchungsformen, vielmehr lassen sich die klassischen Methoden wie Befragung, Beobachtungsstudien, Fallstudien und kontrollierte Experimente darin gut integrieren.

Das prinzipielle Vorgehen bei der Aktionsforschung ist in Abbildung 1 verbildlicht. Im Schritt *Diagnosing* gehen die Forschenden von den real existierenden Fragestellungen der Entwickler und Entwicklerinnen (im Folgenden *Partner* genannt) in deren eigenem Umfeld aus, die mit Hilfe einer Visualisierung potentiell beantwortet werden können und die auch für die Forschenden von wissenschaftlicher Relevanz sind. Hierzu erheben die Forschenden mit den Partnern – etwa durch Befragung oder Beobachtungen – die Anforderungen in Bezug auf die notwendige Art von Information für die Partner, damit diese ihre realen Aufgaben erledigen können.

Im Schritt *Planning Action* erarbeiten die Forschenden mit ihren Partnern genauer, wie man die im Schritt *Diagnosing* als notwendig identifizierte Information erheben und insbesondere visualisieren kann. Die im gemeinsamen Design-Prozess entwickelten initialen Ideen werden noch in diesem Schritt mit Hilfe von evolutionären Prototypen umgesetzt und erprobt.

In der Phase *Taking Action* setzen die Partner die Visualisierung im Rahmen ihrer normalen Entwicklungstätigkeit an ihrer eigenen Software und im eigenen Arbeitsumfeld ein. In dieser Phase werden Daten gesammelt, die geeignet sind, die im Zyklus zu beantwortende Fragestellung zu adressieren. Die genaue Erhebung der Daten hängt von der Fragestellung des Zyklus ab. Sie kann von automatisiert erhobenen Gebrauchsdaten oder Beobachtungen, über Fragebögen bis hin zu kontrollierten Experimenten reichen, abhängig von der Art der Frage und des aktuellen Kenntnisstands.

Im letzten Schritt eines Zyklus werden die gesammelten Daten von den Forschenden ausgewertet und mit den Partnern diskutiert. Hierzu kann auf eine reichhaltige und detaillierte Menge sowohl quantitativer als auch qualitativer Daten aus dem vorherigen Schritt zurückgegriffen werden. Da es die Aktionsforschung im Unterschied zu klassischen kontrollierten Experimenten zur Evaluation von Software-Visualisierung nicht bei punktuellen Erhebungen unter doch eher künstlichen Laborbedingungen belässt, sondern vielmehr sowohl quantitative als auch qualitative Daten über einen langen Zeitraum im realen Umfeld und Arbeitskontext erhebt, kann man sich einen höheren Erkenntnisgewinn erhoffen. Insbesondere ist es auf diese Weise auch möglich, Nachhaltigkeit und Veränderungen über die Zeit zu betrachten, weil die Partner über einen langen Zeitraum begleitet werden. Graduell werden die gewonnenen Erkenntnisse zur Theoriebildung beitragen und damit die Fragestellungen nachfolgender Zyklen mitbestimmen. Teil des letzten Schritts eines Zyklus ist auch eine Reflexion über die gesammelten Erfahrungen in Bezug auf den Forschungsprozess selbst, wie etwa die eingesetzten Instrumente zur Datenerhebung, die Güte und Aussagekraft der gesammelten Daten oder die Zusammenarbeit mit den Partnern, um auf diese Weise zukünftig auch den Forschungsprozess weiter zu verbessern.

III. Fazit

Dieser Beitrag möchte die Aktionsforschung weiter bekanntmachen und zur Diskussion anregen, inwieweit sie sich als empirische Methode in der Forschung zur Software-Visualisierung eignet. Eine Herausforderungen ist es, wie auch bei Fallstudien und Experimenten, die Repräsentativität und Verallgemeinerbarkeit sicher zu stellen. In späteren Phasen müssen deshalb auch Untersuchungen mit anderen Organisationen durchgeführt werden, die nicht primär an der Entwicklung beteiligt waren. Die Chance ist aber höher, dass solche anderen Organisationen zur Teilnahme bewegt werden können, wenn die Visualisierung bereits eine hohe Reife hat.

Referenzen

[1] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, “A systematic literature review of software visualization evaluation,” *Journal of Systems and Software*, vol. 144, 06 2018.

[2] M. Staron, *Action Research in Software Engineering: Theory and Applications*. Springer International Publishing, 2020.