DISS. ETH Nr. 21653

A COMPLEX SYSTEMS APPROACH TO SOFTWARE ENGINEERING

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
MARCELO SERRANO ZANETTI

MASTER OF SCIENCE ETH IN INFORMATIK	ETH ZURICH	CH
ENGENHEIRO ELETRICISTA	UNICAMP	BR
TÉCNICO EM ELETRÔNICA	CTI UNESP	BR

born on JULY 08, 1978

citizen of BRAZIL, ITALY

accepted on the recommendation of
PROF. DR. DR. FRANK SCHWEITZER
PROF. DR. GIUSEPPE VALETTO

2013

DOI 10.3929/ethz-a-010007378

ETH zürich

Abstract

This thesis addresses the complexity of software engineering. We first define what we mean by *complexity*. Then, we proceed by discussing a number of examples of software engineering processes strongly tied to complexity, both of social and technical nature. We specifically focus on two of these processes: *collaborative bug handling* and *software modularity*. The former is the result of social interactions, taking place within open source software communities, while processing and resolving software *bugs*. The latter determines how software architectures should be structured in order to foster their maintainability. This thesis is divided into two parts: one studying the role and dynamics of social interactions, while the other focuses on the growth of source code and how software dependencies are organized. We argue that social interactions and software dependencies share a topological structure that can be addressed quantitatively by the application of *complex networks theory*. We provide a number of examples illustrating how useful this framework can be. We show that we can measure possible threats against community resilience when discussing a case study on centralization. Furthermore, we use social network analysis to create a social information filtering scheme with a remarkable high accuracy, when applied to predict which bug reports can be fixed. We also discuss how the deterioration of modularity can impact on project management costs, and we propose an efficient method to restore software modularity, by automatically reorganizing source code. We argue that the results presented via these two perspectives, namely social interactions and software dependencies, can be unified in software engineering under the *socio-technical congruence* framework. We emphasize that our results have an impact far beyond the realm of software engineering. Specifically, we argue that our findings are of broader interest to the social sciences, contributing to the fast growing field of *computational social sciences*.

Kurzfassung

Die vorliegende Dissertation thematisiert das Problem der Komplexität in der Softwareentwicklung. Als erstes definieren wir, was wir unter Komplexität verstehen. Darauf aufbauend diskutieren wir eine Anzahl von Beispielen aus der Softwareentwicklung, die hohe Komplexität *sozialer* und *technischer* Natur aufweisen. Wir fokussieren uns auf zwei konkrete Prozesse aus der Softwareentwicklung: kollaborative Fehlerbehandlung (*collaborative bug handling*) sowie Softwaremodularität (*software modularity*). Im Fall der kollaborativen Fehlerbehandlung interessieren uns vor allem die sozialen Interaktionen innerhalb einer open source software community, die während der gemeinschaftlichen Behandlung eines Softwarefehlers (*software bug*) stattfinden. Im Fall der Softwaremodularität indessen widmen wir uns der Fragestellung, wie eine Software architektonisch, d.h. in ihrem Aufbau aus Submodulen, gestaltet sein muss, um eine einfache Instandhaltung zu gewährleisten. Diese Arbeit besteht aus zwei Teilen: Teil eins betrachtet die Rolle und Dynamik sozialer Interaktionen, Teil zwei konzentriert sich auf das Wachstum von Softwarecode sowie der Organisation von Softwareabhängigkeiten. Wir legen dar, dass soziale Interaktionen und Softwareabängigkeiten topologische Eigenschaften teilen, die wir quantitativ mit Hilfe der *Theorie komplexer Netzwerke* beschreiben. Anhand einer Reihe von Beispielen zeigen wir die Nützlichkeit dieser Betrachtungsweise auf. Mittels einer Fallstudie zur Zentralität von Entwicklern im sozialen Netzwerk zeigen wir, wie auf diesem Weg mögliche Gefahren für den Zusammenhalt und Erfolg einer Gemeinschaft zur kollaborativen Fehlerbehebung quantifiziert werden können. Diese Erkenntnis verwenden wir dann zur Entwicklung einer Methode, die unter Verwendung Maschinellen Lernens mit erstaunlich hoher Präzision vorhersagt, ob ein software bug behoben werden kann oder nicht. Desweiteren diskutieren wir, welchen Einfluss der Rückgang von Softwaremodularität auf Verwaltungskosten eines Projekts hat, und entwickeln eine effiziente Methode der automatischen Umstrukturierung von Softwarecode um Softwaremodularität wiederherzustellen. Wir argumentieren, dass die Resultate dieser dualen Betrachtungsweise (soziale Interaktion und Softwareabhängigkeit) einen direkten und wichtigen Beitrag zum Forschungsfeld der *sozio-technischen Kongruenz* leisten. Abschliessend möchten wir hervorheben, dass diese Arbeit darüber hinaus unmittelbar Erkenntnisgewinne für weitere Disziplinen liefert, wie zum Beispiel den Sozialwissenschaften, insbesondere für das aktuelle Feld der Computer gestützten Sozialwissenschaften *computational social sciences*.