Preface

As the rate of change and risk of disruption increase relentlessly, companies are constantly battling to proactively adopt new innovations, be it business, technology, or process innovations. No field is more intensely subject to this than the software-intensive systems industry. Ranging from automotive, defense, and telecommunications systems to large, complex installed software solutions, the companies in this industry have been subject to business model innovations, e.g., the transition from products to services; to technology innovations, such as cloud computing and real-time connectivity; and to process innovations, such as agile development practices, continuous integration, and continuous deployment. The challenge for software-intensive systems companies is how to maintain or even improve their competitive position while responding to these disruptions to the normal way of doing things.

Similarly, software engineering research is experiencing its own set of forces in that during the last decades, very few major, industry-changing new innovations have originated in academia. Instead, industry has taken over the role of introducing and driving large-scale adoption of new innovations. For instance, a business model innovation such as open-source software originated in industry. Similarly, technology innovations such as programming languages, ranging from Java to Scala, as well as integrated development environments, such as Eclipse, find their roots in industry. And finally, process innovations such as agile development and continuous integration originate in industry, rather than in academia.

Universities are the homes of numerous highly intelligent, well-trained, and well-intended individuals that are committed to making an impact, and software engineering research groups and departments are no exception. What can then be the reason for the lack of major innovations originating from academia? There are, I believe, three main reasons: First, for a variety of reasons, discussed below, software engineering researchers often have difficulty to gain access to their research environment, which are the large-scale software R&D organizations where software engineering happens. As a consequence, researchers instead focus on small-scale problems that can be studied in a university context, such as studying student projects or otherwise studying simulated, rather than real, environments. Second,

especially over the last decade, the software engineering research community has increasingly demanded empirical data to back up any research claims. This had the intention of reducing the amount of "advocacy research," i.e., researchers presenting claims and providing logically sounding arguments why these claims could be assumed to be true but without any real evidence that the intended outcomes would be seen in reality as well. Although the demand for data accomplished the intended effect, there was an additional effect: software engineering researchers increasingly studied and reported on the current state at software companies as this was the only way to collect relevant data. However, they were no longer innovating on how to improve the current state of practice as the results would not be publishable anyway. Instead, this task increasingly fell to industry. Third, and perhaps most important, academic researchers are not exposed to the market forces experienced by software-intensive systems industries and consequently focus their efforts predominantly on adding more detail, more steps, more activities, more documentation, more intermediate artifacts, more specialization of roles, etc. This focus runs counter on the pressures experienced by industry where the focus is on translating identified customer needs to solutions in the hands of customers as rapidly as possible with as little detail, as few steps and activities, as little or no documentation, and as few artifacts except code as possible, preferably accomplished by anyone who is available for the task at hand. This easily causes a certain level of arrogance among software engineering researchers and a belittling of the accomplishments of numerous outstanding engineers in industry as the goals that these engineers are, consciously or unconsciously, working towards are not properly understood by researchers who project their own goals on industrial practice.

As one may understand from the above, it has proven to be notoriously difficult to build effective, scalable, and long-term software engineering research collaborations between industry and academia. Of course, there are many examples of individual researchers or small groups collaborating for years with a company. And there are examples of companies that have gone out of their way to build relationships with researchers that have lasted for extended periods of time. However, examples of collaborations between sizable groups of relatively diverse software engineering researchers and groups of companies with similar challenges are few and far between. In fact, one of the few long-standing examples of a collaboration of this type is the Fraunhofer Institute for Experimental Software Engineering, and consequently, I am grateful that Professor Dieter Rombach has graciously agreed to provide a foreword for this book.

It was with this understanding of the challenges of collaboration between industry and academia in the area of software engineering that we started the Software Center in 2011. Initially the collaboration started with four founding companies, i.e., Ericsson, AB Volvo, Volvo Car Corporation, and Saab Electronic Defense Systems, and the combined software engineering division between Chalmers University of Technology and Gothenburg University, with three projects and a handful of researchers. Three years later, at the time of writing, we have eight companies and three universities, 15 research projects, and dozens of researchers involved in the Software Center.

Based on the above, it's clear we are on to something. So, what are the mechanisms that have made Software Center successful? There are at least three

basic principles that are worth sharing here: First, all research takes place in 6-month sprints. A sprint starts in January or July and runs for 6 months. During a sprint, the project goes through a full cycle of defining the research problem, designing the research, collecting data or conducting the experiment or trial, analyzing the results and presenting the results to the companies involved, as well as publishing the research outcomes. Each project has a long-term goal and runs for multiple or many sprints, but every sprint results relevant to the companies have to be presented. Second, the technical experts at the companies decide what research is conducted. At the end of every sprint, each ongoing project, as well as each newly proposed project, presents a plan for what to study next. A task force consisting of technical experts at the Software Center companies decides on a ranking of research projects and potentially "kills" projects that are not delivering results relevant to the member companies. This puts an equal balance on academic excellence and industrial relevance. Finally, the longitudinal nature of projects allows researchers to study current state at the member companies, but subsequently to propose improvements. If the improvements are sufficiently appealing, some or all of the software center companies will experiment with the improvement and, if successful, deploy it broadly in the respective companies. This allows software engineering researchers to be involved in and report on improvements in the way software engineering is conducted in world-class companies. The advantage to researchers, obviously, is that it is possible to study more than "current state" as well as the ability to validate innovations at multiple companies, increasing the validity as well as the ease of publication of research conducted in the scope of the Software Center.

Concluding, Software Center is an experiment to establish an effective, scalable, and long-term software engineering research collaboration between academia and industry. The book that you're holding presents the results from the first 3 years. The experiment, so far, is successful in that more companies, universities, and researchers are joining the initiative. Also, many of the results, including the Stairway to Heaven model, the CAFFEA model, the CIViT model, the HYPEX model, as well as many other results, have been adopted or are in the process of being adopted by the partner companies. Finally, over the last 3 years, the partner companies have progressed from experimenting with agile work practices to broad deployment of continuous integration in an agile teams context and the experimentation with continuous deployment of software with selected customers for some companies. As Software Center, our goal is to help companies change faster than without our involvement, and the evidence to date is that we're delivering on that goal.

This book presents the results of the first phase of the Software Center, but it also celebrates the great progress accomplished at the partner companies due to the tireless efforts of the researchers in the Software Center and the champions at partner companies. As director, I am humbled and grateful to everyone involved. All have stepped up to the challenge and actively collaborated to create something that is so much more than the sum of its parts.

Gothenburg, Sweden Jan Bosch
June 2014

http://www.springer.com/978-3-319-11282-4

Continuous Software Engineering
Bosch, J. (Ed.)
2014, XIV, 226 p. 50 illus., 30 illus. in color., Hardcover
ISBN: 978-3-319-11282-4