Impact of the Research Community for the Field of Software Configuration Management

Jacky Estublier (Editor)
Grenoble University
220, rue de la Chimie BP53
38041 Grenoble France
Jacky.Estublier@imag.fr

David Leblang (Co-Editor)
24 Oxbow Road
Wayland, MA 01778
USA
leblang@alum.mit.edu

Geoffrey Clemm
Rational Software
20 Maguire Road
Lexington, MA 02421 USA
Geoffrey.Clemm@rational.com

Reider Conradi
IDI-Gløshaugen
NTNU, NO-7491
Trondheim, Norway
Reidar.Conradi@idi.ntnu.no

André van der Hoek
Dept. of Info & Comp Science
444 Computer Science Bldg
Irvine, CA 92697-3425 USA
andre@ics.uci.edu

Walter Tichy
Informatics U. Karlsruhe
76128 Karlsruhe
Germany
tichy@ira.uka.de

Darcy Wiborg-Weber
Telelogic
9401 Geronimo Road
Irvine CA 92618 USA
darcy@telelogic.com

ABSTRACT
Software configuration management (SCM) is an important discipline in professional software development and maintenance. The importance of SCM has increased as programs have become larger, more complex, and more mission/life-critical.

This paper presents a brief summary of a full report that discusses the evolution of SCM technology from the early days of software development to present, and the specific impact university and industrial research has had along the way.

Full report available at :
http///www-adele.imag.fr/SCMImpact.pdf

Keywords
Software configuration management, software engineering, research impact

1. INTRODUCTION
Software configuration management (SCM) is the discipline of controlling changes in large and complex systems. Its goal is to prevent the chaos caused by numerous corrections, extensions, and adaptations that are applied to any large system over its lifetime. The objective of SCM is to ensure a systematic and traceable development and maintenance process, so that a system is in a well-defined state with accurate specifications and verified quality attributes at all times.

SCM as a discipline has existed for several decades now, and research and industrial advances have shaped the field through numerous contributions. To assess the impact and influence of research on the current state of the art in the field, the authors engaged in an effort leading to the availability of a preliminary version of a full impact report, a brief summary of which is presented here.

ICSE'02, May 19-25, 2002, Orlando, Florida, USA.

During the preparation of the report, the authors engaged in a lively debate about what defines impact and which research to include. We discovered quickly that it was futile to determine who contributed "more", academia or industry. Rather, our goal is to provide an honest and accurate picture of the major research ideas in SCM and show where the ideas have had an impact, irrespective of whether the ideas originated in industry or academia. We hope that the report gives a flavour of the "software configuration management story", a success story that is yet unfolding.

2. HISTORICAL PERSPECTIVE
CM was first developed in the aerospace industry in the 1950s, when production of spacecraft experienced difficulties caused by inadequately documented engineering changes. Software Configuration Management (SCM) is CM tailored to systems, or portions of systems, that consist predominantly of software.

Changes in hardware, software, and business environments and practices have dictated the evolution of SCM since its early days and transformed it from a practice of carefully using differently-colored punch cards to one that routinely employs sophisticated tools that emphasize process support, concurrent engineering control, remote work space control, and other high-level functions. The resulting evolutionary path of SCM has been far from a simple straight line. Some topics have lost their importance, and new problems had to be addressed continuously.

For example, throughout the 1980s, debates focused on determining the best types of compression and retrieval mechanisms. In the 1990s, non-textual objects became more common and totally new algorithms were required. By the year 2000, however, disk storage became so inexpensive and CPUs so fast that compression became unimportant. Many new tools use simple (zip-like) compression.

SCM is now one of the most successful branches of software engineering. A lively international research community is working on SCM and a billion dollar commercial industry has emerged. Nearly all corporate and government software projects use some kind of SCM tool and practitioners consider SCM tools as helpful, mature, and stable.

The success of SCM is not limited to software engineering. Its basic techniques have become pervasive and can be found in many different places. All 4GL tools, most programming environments, and even document editors like Word now include a basic version manager. New domains like web content management also apply SCM techniques, and the new web protocol WebDAV/DeltaV is rapidly gaining acceptance.

3. RESEARCH IMPACT

The success of SCM can be correlated to the key research contributions that have made the transition into industrial SCM products. In particular, industrial products all have as their cornerstones the following four, well-researched issues: versioning and selection; differences, compression, and merging; process support; and distributed and remote development. The report demonstrates how in each of these four areas early research significantly shaped current practice.

Consider, for example, the versioning and selection mechanisms used in high-end SCM systems. Rather than managing individual changes to individual files, these systems have the ability to track logical changes; as a response to a user need to interact with SCM systems at a more natural and process-oriented way. This ability can be traced to the change-sets introduced almost three decades ago by industrials. At that time, the basic versioning mechanism of using revisions and branches (as introduced by SCCS and subsequently improved by RCS) was considered *the* basis for SCM versioning. Change sets, developed in a radically different way by academic research, in practice, were too complex for simple applications, and did not scale well for larger applications. Subsequent industrial research, however, revealed that the change set concept could be streamlined and realized using the standard and efficient RCS like implementation. The resulting change packages concept is now a simple, practical, and manageable solution appreciated by users and compatible with standard versioning.

Research had a similar impact on the other cornerstones. Currently employed storage techniques that integrate compression with delta storage can be traced to research on the topic. Early research integrating general process support into SCM systems, did not succeeded because users found it difficult to define their own processes properly. Nevertheless, all SCM systems currently propose a pallet of best practice processes that now makes SCM one of the few fields in which process is considered an fundamental part and used on a daily basis.

Of note is that often both academic and industrial research is necessary for ideas to succeed in making the transition from research into practice. Sometimes, academic research takes the lead and its ideas can be adopted in a straightforward matter (e.g., compression algorithm). Other times, industrial research is ahead of academia in recognizing and addressing a problem, as was the case with the creation of advanced replication mechanisms to support remote and distributed development. Most often, however, an intricate interplay of research contributions eventually leads to satisfactory solutions. In such cases, the forum provided by the international research community serves as a critical conduit for idea exchange, publication, and discussion (e.g., ICSE, FSE, SCM workshop series).

Of course, SCM research has not just produced successful ideas. Many contributions have not been able to make the transition to industrial practice. The root cause of these failures often lies in the level of complexity required to master the idea, the level of effort required by a customer to use the feature, or the user-perceived lack of actual need. Industry tends to (rightly!) ignore such ideas, until customer practices meet the need for the feature or until a complex idea can be transformed in a way that hides most of the actual complexity for users and implementers.

Some SCM research contributions have found their way into practice in unexpected ways. Smart recompilation, for example, was initially invented to minimize recompilation times by semantically analyzing source code changes to determine the files that needed to be recompiled. Smart recompilation is not found in current SCM tools; however, it is now heavily used in programming environment like Visual Studio.

As can be seen from the above discussion, SCM research has addressed many different topics and it is fair to say that by now the basic principles, concepts, and techniques are set. Although it seems that innovation has slowed down, we consider this situation may only be temporary. Current research is questioning two of the fundamental assumptions that underlie current SCM systems: (a) the focus on managing the implementation of software only and (b) the basic philosophy of SCM being programming language and application independent. Breaking the first assumption requires the management of artifacts produced earlier (e.g., requirements, design) and later in the life cycle (e.g., deployment, dynamic evolution and reconfiguration). Breaking the second assumption involves the integration of SCM functionality into particular environments (PDM, IDE, product lines, architecture). It is clear why these issues are currently being addressed: SCM no longer is a stand-alone discipline. To survive, it needs to continue to stay abreast of new developments, trends, and technologies.

4. CONCLUSIONS

SCM is arguably one of the most successful software engineering disciplines, and it is difficult to imagine this kind of success would have prevailed without research fueling continuous innovations. This report demonstrates that the impact of this research, whether industrial or academic, is undeniable—many of the fundamental techniques underlying current SCM systems were first published in one form or another.

Timing has been critical. Whereas most research contributions were rooted in practical, day-to-day problems, others were too early for their time and/or not practically relevant for the problems at hand. As demonstrated by the remarkable time-delay of change sets, it is often market readiness and/or concept mutation that determines success. Over time, however, most ideas have trickled through in one form or another.

The SCM basic concepts and technologies have been settled, but much work remains to be done. In particular, the field as a whole is now sorting out its relationship to other domains, such as product data management, component-based software engineering, and software architecture. We look forward to the advances that will come from this research, and are proud to be a part of a field with such a rich legacy as SCM.