Journal of Software: Evolution and Process

JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS
*J. Softw.: Evol. and Proc.* 2013; **25**:569–574
Published online 5 July 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.548

Editorial

# A retrospective view of software maintenance and reengineering research – a selection of papers from *European Conference on Software Maintenance and Reengineering 2010*

SUMMARY

As a summary of past, current, and future trends in software maintenance and reengineering research, we give in this editorial a retrospective look from the past 14 years to now. We provide insight on how software maintenance has evolved and on the most important research topics presented in the series of the *European Conference on Software Maintenance and Reengineering*. Copyright © 2011 John Wiley & Sons, Ltd.

## 1. INTRODUCTION

Software maintenance is an active research area because the increasing investments in software and the pressure of customer demands require continuous operations to maintain systems' life as long as possible. Software maintenance in its various forms [1] consumes a significant portion of the software development effort [2] to correct and prevent software errors and to facilitate the evolution of systems over time [3,4].

Traditional and modern development practices (e.g.: agile practices [5]) highlight the importance of software maintenance in order to reduce the amount of technical debt (i.e.: technical debt is a wonderful metaphor coined by Ward Cunningham [6] to describe that systems incur a "debt" caused by previous compromises never satisfied) that has to be repaid.

Today, new software development challenges (e.g.: self-adaptive systems, multiple product development exploiting software variability techniques, service systems engineering, compositional software development, etc.) and the ever-increasing complexity of large software systems require better maintenance techniques and approaches to migrate and reengineer legacy systems to more modern approaches.

As a summary of past, current, and future trends in software maintenance and reengineering research, we give in this editorial a retrospective look from the past 14 years to now. We provide insight on how software maintenance has evolved and on the most important research topics presented in the series of the *European Conference on Software Maintenance and Reengineering (CSMR)*.

## 2. FOURTEEN YEARS OF SOFTWARE MAINTENANCE AND REENGINEERING RESEARCH

The series of the *CSMR* started in 1997 and have provided a valuable and updated state of the art and research trends on software maintenance, reengineering, architecture, and software evolution topics for researchers and practitioners in software engineering. Regarding the evolution of research topics, we performed a retrospective analysis of the main themes for the past 14 years in order to analyze both the origins and the trends of maintenance research. Hence, the following questions arose to our minds:

- *Q1: Which topics have disappeared*?
- *Q2: Which topics have been and are still more popular*?
- *Q3: Which are the emerging trends in software maintenance and evolution research*?

These and other questions can be answered from the data shown in Table I. We grouped papers according to the session names obtained from the table of contents of the *CSMR* proceedings since its first edition. The results are shown in Tables I and II, but papers presented in years 2000 and 2001 (marked with an "*") were not classified under any specific session, so we used the title of the papers to allocate them into the existing categories. In addition, because some of the topics have similar goals, we merged them, as their names may vary a bit along years. Finally, some papers may fit in more than one topic or session, so we classified them in the topic, which seemed more close to the goal of the paper. Table I shows session topics in the early years of software maintenance and evolution research and also those ones that appear only in sporadic years.

Based on the data of Table I, our retrospective analysis for question *Q1: Which topics have disappeared*?, shows that some specific topics were popular only one or two years at most, such as for instance *formal methods, methodology, logic programming, and software configuration management*. The *year 2 K* problem was a popular topic just for only 2 years until it was over, so no further discussion was needed in the following years. Another example is *reuse*, which was a popular topic in the 1990s and less popular afterwards, because many of the reuse processes were assumed under other software engineering activities, which also led to shrinking the number of major Software Reuse conferences from three to one.

In addition, papers belonging to certain topics became isolated as they only appeared for 1 or 2 years but never showed up again. This is the case of papers in the categories *components & objects, dependency analysis, simulation and performance, historical analysis, or aspect orientation*. It is possible that some papers in a popular topic like *clustering*, which only appeared explicitly in 2 years, have been assigned to a different category according to the session names.

Regarding *Q2: Which topics have been and are still more popular*?, we show in Table II those topics that appear more frequently or periodically from 1997 to 2010. These topics indicate that research in these categories keep a continuous record, probably because main research groups in Europe are more active in these areas.

If we observe Table II, topics like *migration, reengineering, system assessment, program understanding, maintenance, and evolution*, appeared quite early but today, there are still hot topics in its various forms, in particular for those papers assigned to broader categories such as *maintenance and evolution*. Others like *refactoring and transformation* have been of interest for a specific period (i.e.: from 2003 to 2006), even if papers including refactoring and program transformations are included in other categories. Surprisingly, *testing* is a popular topic that appears only in 3 sparse years (1999, 2003, and 2006), but we believe more papers on software testing have been published even if they have not been assigned to such category.

Other interesting conclusions can be extracted, such as for instance the *migration* topic, which appears recurrently along years because legacy systems have to be migrated to new technologies, languages, and platforms. Therefore, there are more stable periods where technologies are being used and years where technological changes happen more often, and migration activities are more important. In addition, the last two categories of Table II cannot be considered pure research topics, but some papers have been assigned to *experience reports and tools*. Topics like *maintenance, evolution, program analysis, or static and dynamic analysis* among others can be considered common hot topics for many years in the *CSMR* series.

## 3. CURRENT RESEARCH TRENDS

All in all, the three most populated categories in the history of *CSMR* are, in decreasing order: development, maintenance and effort (42 papers), evolution (36), static analysis and dynamic analysis (27). The view in the last 5 years is consistent, as those three categories are still in the top. As a matter of fact, we could consider these three large issues as the core of the conference topics.

Regarding *Q3: Which are the emerging trends in software maintenance and evolution research*?, it is difficult to deduct from Table II which are the emerging research trends for *CSMR* papers, as many current topics have been also popular in the past. Therefore, we need to look to specific research questions of the papers published to know which topics are more popular today. Among these, we can



DOI: 10.1002/smr

Table I. CSMR topics in early years and less popular ones.

| Topic | 1997 | 1998 | 1999 | 2000* | 2001* | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Methodology* | 4 | | | | | | | | | | | | | | 4 |
| *SCM* | 2 | | | | | | | | | | | | | | 2 |
| *Formal methods* | 3 | | | | | | | | | | | | | | 3 |
| *Logic programming* | | 2 | | | | | | | | | | | | | 2 |
| *Year 2 K* | 5 | 2 | | | | | | | | | | | | | 7 |
| *Management* | | | 1 | | 1 | | | | | | | | | | 2 |
| *Historical analysis* | | | | | | | | 2 | | | | | | | 2 |
| *Clustering* | | | | 1 | | | 3 | | | | | | | | 4 |
| *Information extraction* | | | | | | | | | 3 | | | | | | 3 |
| *Process* | | | | | | | | | 3 | | | | | | 3 |
| *Components & objects* | | | | | | | | | | | 3 | | | | 3 |
| *Dependency analysis* | | | | | | | | | | | 3 | | | | 3 |
| *Quality* | | | | | | | | | | | 3 | | | | 3 |
| *Fault detection & security* | | | | | | | | | | | 3 | | | | 3 |
| *Prediction models* | | | | | 1 | | | | | | | | | 2 | 3 |
| *Simulation and performance* | | | | | 1 | | | | | | | | | | 1 |
| *Reuse* | 2 | 3 | | | 1 | | | | | | | | | | 6 |
| *Components* | | | | | | | 3 | | | | | | | | 3 |
| *Aspect orientation* | | | | | | | 3 | | | | | 3 | | | 6 |
| *Progr. langs* | | | | | | 3 | | | | | | | | | 3 |




DOI: 10.1002/smr

Table II. CSMR most popular topics.

| Topic | 1997 | 1998 | 1999 | 2000* | 2001* | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | Total | 5y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Migration* | 2 | | | | 2 | 3 | | | 3 | 3 | | | 3 | | 16 | 6 |
| *Reengineering* | | 2 | 5 | 1 | | | | | | | | 6 | 3 | | 17 | 9 |
| *System asses.* | | 3 | | 2 | 2 | | | | 3 | 3 | | | | | 13 | 3 |
| *Architecture & prod. families* | | 2 | | | | | | 2 | | | 3 | | 7 | 3 | 17 | 13 |
| *Testing* | | | 1 | | | | 3 | | | 3 | | | | | 7 | 3 |
| *Metrics* | | | | 3 | 1 | | 3 | | | | | | | | 7 | 0 |
| *Architectural understanding & design recovery* | | | 3 | 3 | | 2 | | 3 | | 3 | 3 | | | | 17 | 6 |
| *Development, maintenance & effort* | | 3 | 2 | 1 | 1 | 3 | 6 | 6 | | | 6 | | 5 | 9 | 42 | 20 |
| *Evolution* | | 4 | | 3 | | 3 | | 7 | 3 | | | 6 | 4 | 6 | 36 | 16 |
| *Reverse eng.* | | | | 2 | 2 | 4 | | 3 | | | | 3 | 3 | | 17 | 6 |
| *Refactoring & transformation* | | | | | | | 6 | 3 | | 3 | | | | | 12 | 3 |
| *Web app.* | | | | | 1 | | 4 | 3 | 3 | | 3 | | | | 14 | 3 |
| *Clone & slicing* | | | | 1 | 1 | | 3 | | | | | | | | 5 | 3 |
| *Program analysis, understanding, & comprehension* | 5 | 1 | 2 | | 2 | | | | | 3 | | | | 3 | 16 | 6 |
| *Static- dynamic analysis* | | | 1 | | | 3 | | 4 | 3 | 3 | 3 | 3 | | 7 | 27 | 16 |
| *Feature location* | | | | | 2 | | | | | | 3 | | 3 | | 8 | 6 |
| *Experience reports & empirical studies* | 4 | | | | | | | | 3 | 3 | | 3 | | | 13 | 6 |
| *Tools & frameworks* | | 2 | | | | | 6 | | 7 | 3 | | | | | 18 | 3 |




DOI: 10.1002/smr

mention research on maintenance and evolution of service-oriented architecture (SOA), cloud computing and Web systems, the location of features in software systems that can also be used to describe the variability of product line architectures, the dynamic analysis of programs and runtime issues affecting "self" properties of systems, the evolution of open source software or those perhaps more traditional but always important topics, such as architecture and assessment. The case of reverse engineering seems still important as the migration to modern technologies requires the understanding of current systems, often using reverse engineering, before key refactoring and redesign tasks are applied.

Regarding the papers accepted for this special issue and the current trends in software maintenance and reengineering summarized in Table II, we see a relationship that justifies the popularity of many of the topics shown in the table and what researchers perceive more important based on current problems and technological challenges.

As an example of this, the article **TIDIER: an identifier splitting approach using speech recognition techniques** deals with a word recognition technique and its application to *feature location*, and *CSMR* authors have published six papers in the last 5 years whereas only two papers in previous years. Feature location and feature modeling have become important topics in recent years on behalf of the popularity of software product lines and multiple product development and because much of software requirements are described in terms of system and user features. The article **Incremental reflexion analysis** is about software architecture reconstruction, which according to Table II, we have 17 papers published regularly from years 1999 to 2007. Understanding software systems and architecture reconstruction from source descriptions is a common and recurrent problem for software maintenance as companies need to spend a lot of resources and money in those cases with high-staff turnover. Finally, the article **Realizing service migration in industry – lessons learned** describes a case study (six papers during the last 5 years) which is quite important to show other researchers and practitioners the lessons learned from real maintenance problems. A more detailed description of each of these papers is described in the next section.

## 4. THE ARTICLES IN THIS ISSUE

In the *14th European Conference on Software Maintenance and Reengineering (CSMR 2010)*, which was held on 15–18 March 2010, in Madrid, Spain, we chose the following three best papers for this special issue in the *Journal of Software Maintenance and Evolution: Research and Practice*.

The article **TIDIER: an identifier splitting approach using speech recognition techniques** by *Latifa Guerrouj, Massimiliano Di Penta, Giuliano Antoniol*, and *Yann-Gaël Guéhéneuc* [7] describes a novel approach to recognize words composing source code identifiers, and use this for word recognition technique in continuous speech, and highlights the important role of identifiers in program understanding, as well as traceability recovery and feature location. The approach overcomes the limitations of existing identifier splitting approaches and outperforms previous results by empirical evidence from a case study. Regarding Table II and the topic of the paper, program understanding and feature location are current hot topics, still of major interest today for *CSMR* researchers.

The work of *Rainer Koschke*, **Incremental reflexion analysis** [8], addresses the topic of architecture conformance checking. For this purpose, he uses reflexion analysis to validate the structure of an architecture model against a source model connected by a mapping from source entities onto architecture entities. The author uses reflexion analysis to compute automatically the discrepancies between an architecture model and a source model, and to improve such technique in current tools avoiding recomputation of all the mappings on behalf of incremental reflexion analysis that is carried out only in those parts, which are susceptible of change. Since the 1990s, software architecture checking acquired an enormous popularity as systems increased in their complexity, and today architecture research is an important field for designers and software maintainers.

The third article in this special issue, by *Khalid Adam Nasr, Arie van Deursen*, and *Hans-Gerhard Cross*, is titled **Realizing service migration in industry – lessons learned** [9]. SOA [10] is a hot topic in the software architecture and software development areas, as companies are migrating parts of their software systems to this approach, where parts of the functionalities are engineered as Web services or



DOI: 10.1002/smr

other service technologies (e.g.: peer-to-peer). Hence, the integration of middleware supporting services into current software architectures is a major challenge for SOA development. The authors describe two case studies that deal with reengineering and evolution in the industrial adoption of SOA. The identification of benefits and drawbacks of introducing SOA in large organizations results in key insights for those adopting a similar approach. In addition, interesting research questions focused on measuring the SOA transition, automation of runtime reengineering, and enacting more empirical studies on SOA introduction to achieve progress in the understanding and improvement of development of these systems.

## 5. SUMMARY

Software maintenance in the 21st century is still a big challenge for software companies and developers, as industry needs to reduce software maintenance costs. The diversity of development approaches leads to different maintenance techniques and technologies that are modernized periodically. Conferences such as *CSMR* and others in the field, attempt to present novel approaches aimed to reduce the effort and the burden of maintenance processes, as most of them are still carried out manually. Our retrospective analysis shows that traditional software maintenance research areas are combined with modern topics, like SOA. In other cases, new maintenance techniques and tools, more automation of maintenance tasks, or the application of maintenance processes to recent software development approaches are proposed.

Also, in the past 8 years, more and more experience reports and empirical studies about maintenance have arose, but this area still needs more contributions to show the practical usage of maintenance techniques and their results both in the short and long term. To conclude, the evolution of legacy systems to more modern approaches can be improved and guaranteed on behalf of more research effort in the software maintenance field.

Rafael Capilla
*Universidad Rey Juan Carlos, Madrid, Spain*
E-mail: rafael.capilla@urjc.es

Juan C. Dueñas
*Universidad Politécnica de Madrid, Madrid, Spain*

Rudolf Ferenc
*University of Szeged, Szeged, Hungary*

REFERENCES

1. ISO/IEC 14764:2006. Software engineering – software life cycle processes – maintenance, 2006.
2. Pigosky TM. Practical Software Maintenance. John Wiley & Sons; New York, 1996.
3. Lehman MM, Belady LA. Program Evolution – Processes of Software Change. Academic Press: London, 1985.
4. Madhavji NH, Fernández-Ramil J, Perry DE. Software Evolution and Feedback: Theory and Practice. John Wiley & Sons, 2006.
5. Beck K, *et al*. Manifesto for agile software development. Available at: http://agilemanifesto.org/.
6. Cunningham W. The WyCash portfolio management system. OOPSLA'92, Experience Report, 1992.
7. Guerrouj L, Di Penta M, Antoniol G, Guéhéneuc Y-G. Recognizing word from source code identifiers using speech recognition techniques. *14th Conference on Software Maintenance and Reengineering (CSMR 2010), IEEE DL*, Madrid, Spain, 2010; 69–78.
8. Koschke R. Incremental Reflexion Analysis. *14th Conference on Software Maintenance and Reengineering (CSMR 2010), IEEE DL*, Madrid, Spain, 2010; 1–10.
9. Nasr KA, van Deursen A, Cross HG. Adopting and evaluating service oriented architectures in industry. *14th Conference on Software Maintenance and Reengineering (CSMR 2010), IEEE DL*, Madrid, Spain, 2010; 11–20.
10. Erl T. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005.



DOI: 10.1002/smr