International Journal of Software Engineering
and Knowledge Engineering
Vol. 25, Nos. 9 & 10 (2015) 1715–1720
© World Scientific Publishing Company
DOI: 10.1142/S0218194015710059

The CoCoME Platform: A Research Note on Empirical Studies in Information System Evolution[*]

Robert Heinrich[†,‖], Stefan Gärtner[‡,††], Tom-Michael Hesse[§,§§], Thomas Ruhroth[¶,‖‖], Ralf Reussner[†,**], Kurt Schneider[‡,‡‡], Barbara Paech[§,¶¶] and Jan Jürjens[¶,***]

[†]*Karlsruhe Institute of Technology, Germany*

[‡]*Leibniz Universität Hannover, Germany*

[§]*University of Heidelberg, Germany*

[¶]*TU Dortmund, Germany*

[‖]*heinrich@kit.edu*

[**]*reussner@kit.edu*

[††]*stefan.gaertner@inf.uni-hannover.de*

[‡‡]*kurt.schneider@inf.uni-hannover.de*

[§§]*hesse@informatik.uni-heidelberg.de*

[¶¶]*paech@informatik.uni-heidelberg.de*

[‖‖]*thomas.ruhroth@cs.tu-dortmund.de*

[***]*jan.jurjens@cs.tu-dortmund.de*

Methods for supporting evolution of software-intensive systems are a competitive edge in software engineering as software is often operated over decades. Empirical research is useful to validate the effectiveness of these methods. However, empirical studies on software evolution are rarely comprehensive and hardly replicable. Collaboration may prevent these shortcomings. We designed CoCoMEP — a platform for supporting collaboration in empirical research on software evolution by shared knowledge. We report lessons learned from the application of the platform in a large research programme.

Keywords: Software evolution; empiricism; information system; research platform.

1. Joint Research Facilitates Empirical Studies in Software Evolution

Many information systems are operated over decades while facing various modifications, e.g. due to emerging requirements, bug fixes, and environmental changes. In consequence, the software changes continually which is named software evolution. Supporting software evolution is a competitive advantage in software engineering. Various methods are available to support diverse aspects of software evolution.

[*]This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future — Managed Software Evolution.

However, it is hard to assess the effectiveness of these methods and to compare them due to divergent characteristics. Empirical research in terms of case studies and controlled experiments is useful to validate evolution methods. Yet, empirical studies on software evolution are rarely comprehensive (as further discussed in [1]) since many aspects are needed to study evolution, such as (a) long time-frames of observation, (b) large amount of artifacts, (c) various types of artifacts, and (d) access to relevant project data.

We believe it is essential to collaborate by joint research in order to increase coverage of these aspects. Joint research supports sharing of knowledge and resources [2] which promises to increase study comprehensiveness (e.g. by considering more and heterogeneous artifacts from different sources) and efficiency (e.g. by reusing artifacts or evolution scenarios). Furthermore, joint research supports study replication and confirmation [3] as research is conducted on a common basis such as tool infrastructure, or common data or source code. Our goal is to support collaboration in and replication of empirical studies by joint research based on common evolution scenarios and artifacts. Existing empirical studies on software evolution are seldom comparable as they vary in analyzed subjects and execution process. Further, these studies are rarely reusable as important artifacts (e.g. requirements or context knowledge) are often not provided to the community. To the best of our knowledge, there is neither a community-accepted case study for software evolution nor a common benchmark available. Consequently, a common basis for study collaboration and replication is missing [1].

This research note presents CoCoMEP[a] — a platform for collaborative empirical studies on information system evolution. It gives an overview of the platform originally published in [1]. Under a "platform" we understand a comprehensive knowledge base for empirical research that can be exploited and extended by researchers with different backgrounds and research interests. It provides assistance on diverse characteristics important for software evolution, e.g. the life-cycle of the system, comprehensive evolution scenarios, and artifacts in different revisions (Sec. 2). CoCoMEP is applied for collaboration among several projects within the DFG Priority Programme *Design For Future — Managed Software Evolution* (SPP1593) [5]. These projects gathered lessons learned on research collaboration in software evolution (Sec. 3). CoCoMEP, however, is not limited to SPP1593 but open for reuse and extension by researchers outside the scope of the priority programme.

2. The CoCoME Platform Supports Joint Research by Standardization

We analyzed related work on empirical research in [1] with regard to collaboration. The aim was to learn from experiences and to identify the following requirements as basis for the design of CoCoMEP. *R1*: the case study must be standardized in terms

[a] The term is a combination of Common Component Modeling Example "CoCoME" [4] and "Platform". Additional information on CoCoMEP is available online http://www.dfg-spp1593.de/cocome.

of activities and artifacts. *R2*: it must enable effective collaboration among researchers. *R3*: the case must comprise artifacts that correspond to all life-cycle phases. *R4*: the evolution process must contain iterations and increments. *R5*: the application, problem, and solution domains of the case must be defined, e.g. by using natural language text or models. *R6*: tools necessary to replicate the case must be evaluated.

In a literature review in [1] we examined existing empirical studies on software evolution. We could not find a study considering the entire evolution life-cycle. In addition, neither artifacts nor relations between the different development activities are comprehensively covered by existing studies. Most empirical studies and their results are not comparable in terms of domain, size, or complexity. Thus, obtained results have limited evidence.

According to the requirements, we developed the research platform CoCoMEP depicted in Fig. 1. On this account, the established CoCoME system [4] serves as the study subject. We developed examples of change scenarios in information system evolution, constructed sample activities in system development and operation, and arranged them in life-cycle form. The interconnected parts of CoCoMEP are explained in the following.

An **Evolution Subject** is the amount of artifacts in different revisions that represent an information system. CoCoME has been set up in a Dagstuhl research seminar as a community case study for component-based software engineering. We evolved CoCoME to a study subject on which methods in the context of software evolution are applied. CoCoME resembles a trading system of a supermarket chain handling sales. The system implements sales processes at a single store of the chain, e.g. scanning products or paying, as well as enterprise-wide administrative tasks, e.g. inventory management. Figure 2 gives an overview of the CoCoME system by illustrating its use cases. CoCoME in general as a community case study balances real-world relevance with suitability for an academic environment. It enables comparison between different software modeling and analysis approaches and identification of limitations in software evolution research. Detailed description of the CoCoME architecture by component-, deployment-, and sequence diagrams is given in [4]. Several variants of CoCoME exist that span different platforms and technologies. These range from plain Java code and service-oriented frameworks to hybrid cloud architectures. Furthermore, various development artifacts are available, such as

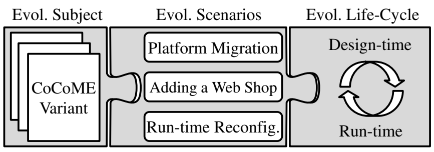

Fig. 1. Overview of the CoCoME platform.

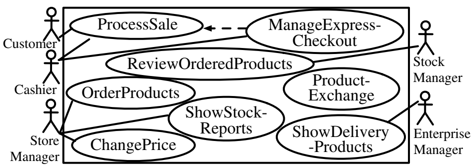

Fig. 2. Overview of the CoCoME use cases.

requirements specification or design documentation, which changed over time. CoCoME is well suited to serve as evolution subject because the supermarket context is commonly comprehensible and the complexity of the system is appropriate. As CoCoME is a distributed system, several quality properties are affected by evolution.

An **Evolution Scenario** describes changes to a certain evolution subject. Based on CoCoME, we implemented distinct evolution scenarios (S1-S3).

S1: Web Shop Extension: A web shop is added where the customers can order online and pick-up the goods at a chosen store. This design-time modification includes adding new use cases and modifying existing design models. S1 transforms a closed system (only employees can access) to an open system (customers can accessed via internet). Hence, various quality properties are affected, e.g. privacy, security, performance, and reliability.

S2: Platform Migration: The enterprise server of the trading system and its database are now running in a cloud environment to reduce operating costs of resources. The introduction of the cloud enables flexible adaptation and reconfiguration of the system, however, causes new challenges regarding aforementioned quality properties.

S3: Database Migration: During a big advertise campaign, the performance of the system may suffer due to limited capacities of the current cloud provider. Migrating the database from one cloud provider to another may solve the scalability issues, however, may cause privacy issues. In [6] we sketch privacy-relevant changes in the cloud context.

An **Evolution Life-Cycle** integrates activities and their relationships required to implement evolution scenarios. We developed a set of sample activities typical in information system evolution and arranged them in life-cycle form (cf. process model in Fig. 3) to cope with aforementioned evolution scenarios. An iteration in the life-cycle starts with a change request, e.g. for S1 or S2. Emerging requirements are elicited and documented. Decisions are made and documented. A static quality analysis is conducted to identify quality issues at design-time. The design is adapted, if necessary, and implemented. After deployment, a dynamic quality analysis is conducted for the running system to identify run-time issues which may result in automated adaptation (S3) or a new iteration for manual evolution.

Diverse variants of the three parts of CoCoMEP are possible. In principle, CoCoMEP is appropriate to conduct empirical studies on software evolution as it satisfies the aforementioned requirements. *R1*: CoCoMEP provides standardized study subject, evolution scenarios, and life-cycle activities. *R2*: this standardization

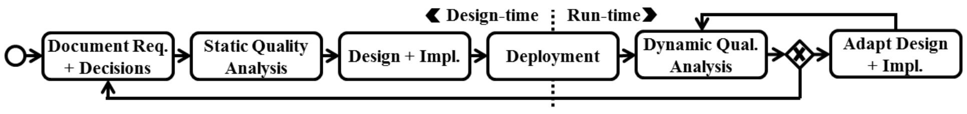

Fig. 3. Overview of the Evolution Life-Cycle applied in the DFG Priority Programme SPP1593.

of the research platform in conjunction with the community offers a structure for collaboration and study replication (see Sec 3). *R3*: CoCoMEP comprises activities and artifacts that correspond to all phases in the system's life-cycle. *R4*: it covers iterations and increments in the development process. *R5*: it provides a concrete setting to define the application domain (i.e. supermarket chain), problem domain (i.e. web-based system) and solution domain (e.g. architecture, code, etc.) of the case. *R6*: it supports evaluating the tools necessary to replicate the case, such as implementation/design languages, operating system, or development environments.

3. Applying the Platform Contributed to Collaboration among Researchers

CoCoMEP targets researchers dealing with empirical studies on modeling or analysis approaches in the software evolution context who want to utilize collaboration and replication capabilities of a community case study and thus increase community acceptance.

In this section, we discuss an excerpt of the most important lessons learned from applying CoCoMEP in SPP1593 to give an impression of its use and effectiveness. We list benefits perceived while applying CoCoMEP and potentials for improvement. CoCoMEP proved to be an appropriate knowledge base and supported us in: (i) *Gathering project-spanning understanding.* Mapping the diverse development and operation activities and artifacts specific to the single projects within SPP1593 into the given life-cycle structure enabled a common understanding of them. Further, common understanding has been supported by a joint communication and documentation infrastructure, i.e. mailing lists, media wiki, and SVN repository. The wiki contains all the information about life-cycle activities and related artifacts to be shared. We use the SVN repository to share source code as well as configuration and documentation artifacts. Based on the life-cycle and infrastructure it was easy to identify and solve uncertainties and misunderstandings among participants from different projects and to create a project-spanning understanding. This is a necessary foundation for research collaboration. (ii) *Identifying common artifacts.* Mapping activities and artifacts into the life-cycle allows for identifying artifacts used by diverse projects and relations between artifacts. This is another foundation for research collaborations. (iii) *Reuse.* The life-cycle also allows for reusing activities and artifacts among the projects. On the one hand, some activities are performed by multiple projects. On the other hand, the output (i.e. artifacts) of activities associated to one project is often reused as an input for activities associated to another project. This contributes to efficiency and the evaluation of the artifacts and thus the applied approaches. (iv) *Clarifying interfaces between projects.* Project-spanning understanding and knowledge about dependencies between activities and artifacts supports clarifying the interfaces between the single projects. This leads to distribution of responsibilities and thus results in more efficient collaborations. (v) *Establishing a technical basis.* CoCoMEP contributed to the development of a common technical basis between the single projects. It supported us in developing tools that interact with each other based on clearly defined interfaces and in configuring common execution environments which reduces effort for the single projects and eases collaboration and study replication.

Applying CoCoMEP in the SPP1593 context, however, showed some potentials for improvement. Change history of the artifacts is rather short. Since SPP1593 started in 2012, artifacts still face few evolutionary changes compared to ordinary repository mining studies for instance. This is caused by the fact that CoCoME is a research prototype and we do not have the amount of human and financial resources involved in real-life development. Nevertheless, as shown by studies in SPP1593, CoCoME provides a sufficient knowledge base so far for conducting various analysis, e.g. on use cases, decisions, or monitoring and simulation data. We are confident to produce a larger change history in the future as the priority programme continues for three more years and simultaneously the CoCoME system is applied in a growing number of studies beyond the programme.

4. Conclusion

Based on requirements for collaboration support from related work and a literature review on empirical studies on software evolution, we developed CoCoMEP. The platform consists of three interconnected parts — an established evolution subject, related evolution scenarios, and a life-cycle covering activities to address the scenarios. Thus, it supports collaboration in and replication of empirical studies as perceived while applying CoCoMEP in a large research programme. In the future, the subject CoCoME will be further evolved by new scenarios which may include parallel evolution and co-evolution of artifacts.

References

1. R. Heinrich *et al.*, A platform for empirical research on information system evolution, in *27th Int. Conf. on Software Engineering and Knowledge Engineering*, KSI, 2015, pp. 415–420.
2. D. I. Sjoberg *et al.*, The future of empirical methods in software engineering research, in *Future of Software Engineering*, IEEE, 2007, pp. 358–378.
3. N. Juristo and O. Gómez, Replication of software engineering experiments, *Empirical Software Engineering and Verification*, 2012, pp. 60–88.
4. S. Herold *et al.*, CoCoME — The common component modeling example, in *The Common Component Modeling Example* (Springer, 2008), pp. 16–53.
5. U. Goltz *et al.*, Design for future: Managed software evolution, *CSRD*, 2014, pp. 1–11.
6. R. Heinrich *et al.*, Integrating run-time observations and design component models for cloud system analysis, in *MRT*, CEUR Vol. 1270, 2014, pp. 41–46.