

Empirical Studies of Software Maintenance: A Report from WESS '97

Lionel Briand	Filippo Lanubile	Shari L. Pfleeger	Gregg Rothermel	Norman Schneidewind
Fraunhofer IESE, Germany (workshop co- chair and group chair)	University of Bari, Italy (workshop co- chair)	Systems/Software, Inc., USA (group chair)	Oregon State University, USA (group chair)	Naval Postgraduate School, USA (group chair)

The Second International Workshop on Empirical Studies of Software Maintenance (WESS '97) hosted by the International Conference on Software Maintenance (ICSM '97), took place October 3rd, 1997 in Bari, Italy. Over 50 participants gathered to discuss and better understand the way we perform software maintenance.

The philosophy of this workshop is that a true understanding of software maintenance issues can be achieved through empirical studies, i.e., studies which provide evidence based on empirical data collected in a disciplined manner. Unfortunately, the empirical approach is not yet a mainstream practice in the software maintenance field and we still find advocates of new methods or tools whose claims are based on merely logical reasoning or anecdotes of personal use.

The theme of this second edition of the workshop was "Towards Collaborative Projects". The aim was to encourage researchers and practitioners to collaborate on mutually profitable projects and thus incrementally build a robust body of knowledge on software maintenance, and more generally on software evolution. This workshop provided a forum for empiricists to discuss what are the key issues, share results from past studies, review study designs, and call for participation in new projects.

WESS '97 was organized as four parallel interactive working groups: (1) defect detection and analysis, (2) program analysis, impact analysis, and testing, (3) maintainability of object-oriented systems, and (4) maintenance practices within and across organizations. Attendees in each group presented key issues from position papers submitted to the workshop. Then, discussions took place to identify common important issues and to address questions such as: "what we already know?", "what we need to know?", "how we can proceed and collaborate to know more?". The results of the four working groups were presented in a final general session, and are summarized in this report.

Working Group 1: Defect detection and analysis

Chair: Norman Schneidewind, Naval Postgraduate School, USA

Participants:

Khaled El Emam, Fraunhofer IESE, Germany

David Fuschi, SIA, Italy

Niclas Ohlsson, Linköping University, Sweden

Carolyn Seaman, University of Maryland, USA

Forrest Shull, University of Maryland, USA

Claes Wohlin, Lund University, Sweden

The following issues emerged from presentations and discussions:

1. We discussed the problem of comparability of research studies across different application, development, and maintenance environments. For example, can defects be compared across organizational and application boundaries? A partial solution to this problem would be for researchers to differentiate research results that are generally applicable from those that are limited in applicability to a specific operating environment. In addition, the use of industrial classification codes, which classify industries by type, could be noted in the research results.
2. We identified the difficulty of tracing defects found during inspection to subsequent faults and failures experienced during testing and operation. It is difficult to make accurate product reliability predictions based on early defect discovery, although there has been some progress in using software metrics to predict fault proneness of software. Defects are subject to different interpretations depending on the phase in which they were introduced and discovered. For example, a defect in a requirements specification during requirements analysis and undetected until the operations phase could have disastrous consequences compared to a coding error made during the coding phase and discovered during testing. There is the need to record detailed context information about documentation, insertion phase, and discovery phase in the defect database.
3. We considered whether the validity of statistical methodology in metrics analysis should be compromised for the purpose of making the metrics results understandable to practitioners. A consensus was not achieved. However there was considerable discussion about the implications of this approach. Some participants felt that this approach should not go so far as to invalidate the conclusions of the metrics analysis. An alternative to diluting the statistical analysis is for researchers to provide training for practitioners in statistical methods.
4. We agreed that there should be more research in the important area of the interaction between hardware and software as a cause of faults and failures. Historical problems in this area have been: 1) difficulty in determining and recording whether a failure was caused by hardware, software, or both; 2) lack of integration of hardware and software in reliability models and predictions (i.e., lack of system models and predictions); and 3) lack of communication and

cooperation between hardware and software engineering groups. The technical problems in this area could be addressed by early implementation in the development process of instrumentation to monitor both hardware and software faults and failures and to correlate the two.

Working Group 2: Program Analysis, Impact Analysis, and Testing

Chair: Gregg Rothermel, Oregon State University, USA

Participants:

Mary Jean Harrold, Ohio State University, USA

Catherine Jaktman, University of Technology Sydney, Australia

John Kyaruzi, University of Dar es Salaam, Tanzania

Bruno Lague, Bell Canada, Canada

Minna Makarainen, VTT, Finland

Michele Paradiso, IBM SEMEA Sud, Italy

Adam Porter, University of Maryland USA

Tom Reps, University of Wisconsin, USA

David Rosenblum, University of California, Irvine, USA

Tomeki Tolliver, Howard University, USA

In recent years there has been increasing interest in the application of program analysis techniques to problems in software maintenance and testing. This working group focused primarily on the following four issues relating to empirical studies of such techniques.

1. In general, we agreed that existing infrastructure for building analysis tools is inadequate. However, efforts to build such infrastructure face serious obstacles, such as dependence on a variety of program-analysis tools that process real source code. In general, it is difficult and time-consuming to construct such tools, and adequately documented, extensible, publicly-available versions of such tools are not available. We agreed that we need to more carefully construct interfaces to our analysis tool components to facilitate the production of new tools. We also agreed that it is worth investigating the feasibility of using standard databases to store intermediate language outputs and tool outputs; these databases could then serve as input points to provide data for other analysis tools. We also need to watch for opportunities to contribute to projects (such as NSF projects) that may, given appropriate forethought, be able to provide prerequisite infrastructure.
2. The subjects for experiments with program-analysis techniques are primarily "software artifacts" rather than humans. This lets us avoid several problems inherent in human studies, but raises problems of its own -- in particular, problems of obtaining sufficient experimental subjects. Depending on the techniques being studied, we may require various programs, versions of those programs, input distributions, test suites, fault data, and documentation. Free software seldom includes all of these artifacts; commercial software vendors, who are more likely to maintain such artifacts, are often reluctant to make them available, or allow researchers to publish potentially "negative" results about them. One participant reported an experience in which another researcher had refused to provide artifacts described in a conference publication "until the journal version appears."

The participants therefore agreed that we need to build repositories of experimental subjects, and make these available to other researchers.

3. We need to develop better cost/benefits models and a greater understanding of what data is important to collect. Such models themselves require validation, but once accepted, could facilitate comparative studies. Also, we need to develop benchmark suites of subjects, and to do this, we need to find ways to characterize workloads. We agreed that simulation techniques that provide useful views of costs/benefits may suffice in the absence of rugged prototypes. In general, we should use experimentation proactively to drive research, not just to validate or compare tools, and we should seek ways to determine empirically whether approaches are more or less likely to result in high/low payoff, prior to full investment in tool development.
4. Many of the analysis-based techniques proposed to date have focused on single procedures, and been investigated only in application to small subjects. We must investigate scalability issues. Some general strategies that we discussed, many of which have also been suggested in the literature, include the use of demand-driven algorithms, persistent caching of data to avoid recomputation; coarse-grained analyses that efficiently compute summary information about a program; and separate analyses that analyze program components individually and then reuse the analysis information in later analyses. Techniques for utilizing databases and database technology should be considered. Techniques that combine dynamic and static analyses also appear promising. Finally, we need to team up with experts in visualization, to investigate techniques for representing analysis information usefully.

Working Group 3: Maintainability of Object-Oriented Systems

Chair: Lionel Briand, Fraunhofer IESE, Germany

Participants:

Maria Pia Angelini, Corinto, Italy

Giuliano Antoniol, IRST, Italy

Jim Buckley

John Daly, Fraunhofer IESE, Germany

Christian Douce, UMIST, UK

Antonietta Guerra, Corinto, Italy

Mikael Lindvall, University of Linköping, Sweden

James Miller, University of Strathclyde, UK

Maria Nella Palese, Corinto, Italy

Daniel Proulx, Bell Canada, Canada

Vaclav Rajlich, Wayne State University, USA

Magnus Runesson, University of Linköping, Sweden

The main objective of this group was to investigate the factors affecting the maintainability of Object-Oriented (OO) systems, identify the various dimensions of maintainability and discuss how to measure them.

Four dimensions were identified as being relevant for maintainability:

1. Comprehensibility: the ease of understanding a system or a part thereof.
2. Ease of impact analysis: the ease to identify the system parts to be modified when a change to the system has been requested.
3. Localisability: the extent to which changes to the system tend to be confined to a small part of it.
4. Testability: the ease of testing the system, i.e., effort versus coverage.

A number of factors strongly affecting maintainability were identified. There were classified in three categories: product factors, environment factors, and people factors. The focus of the discussion was on the issue that was deemed the most difficult, that is one of the product factor referred to as design quality. Some of the relevant dimensions of design quality that were deemed significant, based on the group members experience, were:

- class coupling and cohesion,
- pattern usage and conformance,
- inheritance depth,
- multiple inheritance usage,
- degree of implementation inheritance,
- degree of COTS use and reuse, and
- misuses of polymorphism.

The main maintenance problems that were specifically associated with Object-Oriented were that

- inheritance leads to concept assignment problems in the lower levels, i.e., the difficulty to assign a clear concept in the problem domain to a class in the design; and
- the OO class structure has a tendency to delocalize the system functions, i.e., different parts of the function are implemented across several classes in the design.

The research agenda that was identified was, at a high level, threefold:

1. The relationships between the design factors identified above and the various maintainability dimensions need to be investigated across multiple environments.
2. The benefits and drawbacks of Object-Oriented must be investigated from a maintainability perspective, which has been so far a neglected quality aspect.
3. We need to learn how to better measure these various maintainability factors and dimensions.

Object-Oriented (OO) has had a very significant impact on what is perceived as maintainable in design and architectures. The implications of OO in that respect have not yet been fully understood. The software engineering community needs to investigate what makes OO systems maintainable and what maintenance pitfalls must be avoided when using OO design and architecture. Since no analytical reasoning can fully answer these questions, answers must be sought through well designed empirical studies and rigorous measurement.

Working Group 4: Maintenance Practices Within and Across Organizations

Chair: Shari Lawrence Pfleeger, Systems/Software, Inc., USA

Participants:

Vic Basili, University of Maryland, USA
Anna Rita Fasolino, University of Naples, Italy
Filippo Lanubile, University of Bari, Italy
Timothy Lethbridge, University of Ottawa, Canada
Loredana Mancini, O.Group, Italy
Bernard Moreau, France Telecom, France
Elmar Pritsch, Ericsson, Sweden
Dieter Rombach, University of Kaiserslautern, Fraunhofer IESE, Germany
Jarrett Rosenberg, SUN Microsystems, USA
Susan Sim, University of Toronto, Canada
Anders Subotic, Linköping University, Sweden
Ray Toal, Dublin Bank of Ireland, Ireland
Van Solingen, Schlumberger RPS, The Netherlands
Larry Votta, Bell Labs, Lucent Technologies, USA
Nicholas Zvegintzov, Software Management Network

This working group represented many different perspectives. Some participants were interested in tool use: what is used? what difference do the tools make to maintenance effectiveness and efficiency? Others wanted to investigate actual practice: what do maintainers do, and where is there room for improvement? Still others wanted to build decision support tools and models to renew, reengineer or replace legacy systems. Each participant focused on a subset of the characteristics of maintenance organizations, determined to a large degree on business needs and pressures. Some offered data for analysis; others were grateful for data to replicate experiments and assess statistically.

As we discussed the overall topic, it became clear to all of us that we need to take several steps in understanding what we do and how to improve it.

1. We must understand the domain in which we are building and maintaining software.
2. We focus on understanding individual pieces of the domain,
3. We focus on understanding the relationships among the pieces
4. We devise and implement intervention strategies to improve our processes, products and resources.

Using these steps, it is easier to notice that some aspects of maintenance are more mature than others.

We applied the four-step model to various aspects of software maintenance: products, process, people and organizations, each in the context of customers, environment and contextual process drivers.

- For products, we want to know what products are out there, including the services and functions they provide. Then, we want to know the demographics of those products, including attributes and relationships. That is, we want to investigate the nature of faults and failures, the degree to which we can trace from one

development product to another, the localization of information within a product, the infrastructure used to support products, and the overall software architecture of legacy systems. Within each product, we need to know what persists, what changes and evolves, and what eventually is discarded. This information lays the groundwork for how to improve our products and how to anticipate what is likely to change.

- Similar information is needed about processes: what activities do we perform, and what is the goal of each activity? In addition, we must understand what techniques are used (and what are NOT used) and why.
- Maintainers use skills that overlap with but are not the same as those of developers. Thus, we should study what kinds of people do maintenance, and what kinds of activities they perform. What training do they need? With what are they experienced? What aspects of their education and background are relevant to performing maintenance effectively?
- Finally, we need information about maintenance organizations: their size, the experience each team has in working together, and the organizational constraints that affect the way they do their jobs. We also need to know the communication paths, the demographics of the team structure, and the turnover as people leave to work on other projects.

Summary

This workshop has built a basis for documenting what we know about maintenance. We hope to build on this framework in future WESS workshops, so that we eventually build a body of evidence about how to make maintenance more effective and efficient. We envision the way forward involving two major thrusts: looking at other disciplines, and collaborating within our own discipline. Engineering, business management, psychology, and the social sciences can teach us how similar problems have been solved in the past. We can learn techniques of organization, communication and assessment that will be useful in understanding what we do and how we can do it better. Collaboration should involve finding volunteers to analyze common problems, and coordinating several kinds of studies to build compelling evidence that some techniques are more useful than others. As we perform our investigations, we should document the next steps, and describe our study's limitations so that we can improve the way we seek knowledge as well as the knowledge we seek.

Additional Information

The proceedings of WESS '97 [1] contain the position papers in their final version. A list of position papers as well working group reports are available electronically at the workshop web site:

<http://www.cs.umd.edu/~lanubile/wess97/>

The next workshop, WESS '98, will be chaired by Shari Lawrence Pfleeger (Systems/Software, Inc., email: s.pfleeger@ieee.org) and Jarrett Rosenberg (SUN Microsystems, email: Rosenberg@Eng.Sun.com).

WESS '98 is planned to occur on November 16th, at Bethesda, Maryland, in conjunction with ICSM '98 and METRICS '98.

References

- [1] L. Briand and F. Lanubile (eds.), *Proc. of the 2nd International Workshop on Empirical Studies of Software Maintenance (WESS'97)*, ISBN 3-00-002030-6, 1997.