Empir Software Eng (2007) 12:107–119
DOI 10.1007/s10664-006-9030-4

Protocols in the use of empirical software engineering artifacts

Victor R. Basili · Marvin V. Zelkowitz · Dag I. K. Sjøberg · Philip Johnson · Anthony J. Cowling

Published online: 5 December 2006

Editor: Lionel Briand

Abstract If empirical software engineering is to grow as a valid scientific endeavor, the ability to acquire, use, share, and compare data collected from a variety of sources must be encouraged. This is necessary to validate the formal models being developed within computer science. However, within the empirical software engineering community this has not been easily accomplished. This paper analyses experiences from a number of projects, and defines the issues, which include the following: (1) How should data, testbeds, and artifacts be shared? (2) What limits should be placed on who can use them and how? How does one limit potential misuse? (3) What is the appropriate way to credit the organization and individual that spent the effort collecting the data, developing the testbed, and building the artifact? (4) Once shared, who owns the evolved asset? As a solution to these issues, the paper proposes a framework for an empirical software engineering artifact agreement. Such an agreement is intended to address the needs for both creator and user of such artifacts and should foster a market in making available and using such artifacts. If this framework for

V. R. Basili · M. V. Zelkowitz (✉)
Department Of Computer Science, University of Maryland,
AV Williams Building, College Park, MD 20742, USA
e-mail: mvz@cs.umd.edu

V. R. Basili
e-mail: basili@cs.umd.edu

V. R. Basili · M. V. Zelkowitz
Fraunhofer Center Maryland, College Park, MD, USA

D. I. K. Sjøberg
Simula Research Laboratory, Lysaker, Norway
e-mail: dagsj@simula.no

P. Johnson
University of Hawaii, Honolulu, HI, USA
e-mail: Johnson@hawaii.edu

A. J. Cowling
University of Sheffield, Sheffield, UK
e-mail: A.J.Cowling@dcs.shef.ac.uk

sharing software engineering artifacts is commonly accepted, it should encourage artifact owners to make the artifacts accessible to others (gaining credit is more likely and misuse is less likely). It may be easier for other researchers to request artifacts since there will be a well-defined protocol for how to deal with relevant matters.

Keywords Data sharing · Experimentation · Ownership of data · Proper use and reuse of data and artifacts · Testbeds

1 Introduction

Empirical study involves the generation of artifacts (i.e., data, metadata and schema for the data, as well as source code and other development objects) and the use of experimental testbeds of various kinds. These involve effort and cost on the part of the group who developed them. The developers of these items would like to share them with the community for a number of reasons (e.g., to contribute to the replication of experiments, to allow meta-analysis, to get more people involved in the empirical research). On the other hand, they would like an acknowledgement of their original effort as well as some control of the use of these artifacts in order to know how they are being used, to prevent misuse, and to ensure that the results of any work using these items is available. It is also worth noting that there is a cost to maintain these items, to interact with users who may have questions about these items and to track their evolution. How such items could be shared in a way that is equitable to all parties is the focus of this paper.

One extreme is to allow anyone who wants to use the items to be able to use them at their own risk. This may incur a cost on the part of the artifact owner, if the artifact owner wishes to cooperate. The other end of the spectrum is to limit access to only those groups that are working directly with the original developer of these items.

Artifacts are characterized by the set of explicit characteristics of the environment in which they are created. Over time, users of an artifact may change it or add features (e.g., new data analysis, new documents related to the artifact). Therefore, artifacts evolve and different versions may not have the same meaning or relevance in new contexts. If different groups separately alter artifacts derived from the same source, the resultant properties of each artifact may differ. How does one provide for the provenance of the artifacts so that such alterations can be tracked and understood?

2 Background

2.1 Software Engineering Experience

Within the empirical software engineering research community, the authors have experience with several examples of testbeds, experimental data and other artifacts that have been used by several research groups. This experience has now reached the point where it is appropriate to apply principles of empirical software engineering to it, in order to analyse the issues that arise from how these research groups shared data and what problems they had. The following descriptions do not explain these projects in depth, but just address typical data sharing and ownership issues. The analysis of these examples was then used to develop a plan to allow for the effective sharing of information across the entire empirical

software engineering research domain. The significant issues that arose were: (1) What were the goals of the project? (2) What kinds of data were collected or artifacts developed? (3) How many users were there, if known? And, most important for this analysis, (4) What were the primary problems associated with the data ownership and how were they dealt with?

(1) *NASA Goddard Space Flight Center Software Engineering Laboratory (SEL):* The NASA SEL from 1976 to 2002 studied software development in the Flight Dynamics Division at NASA/GSFC. The SEL collected data on the resources expended, changes and defects, product characteristics, and processes applied as well the conformance to those processes on several hundred projects. Various project characteristics were also collected, e.g., context variables. Models of cost, schedule, and quality were built using this data. The artifacts developed were operational NASA systems including source code as well as the various requirements, design, and testing documents (Basili et al. 2002).

For years the data was given away freely and artifacts were shared with whoever asked for them. But: (1) the data was often misused or misinterpreted, (2) the results of the analysis were often not known or shown to the original data owners, (3) the SEL had to maintain this repository and had to spend time in organizing and making the data available to requesters, and (4) the opportunities for feedback and meta-analysis were often lost. With the closure of the SEL in 2002, item (3) in the list meant that much of the valuable data has been lost.

(2) *University of Maryland software analysis projects.* In the mid-1980s, multiple organizations ran an experiment to evaluate the effectiveness of reading as a defect detection technology. Major foci included the development of techniques for reading requirements documents and object oriented designs. To run these studies several artifacts (e.g., fault seeded code documents or requirements documents, data collected on the defects detected) were generated. These were given to several researchers for their own replication.

Although experiments were replicated, the experience of the experimenters was that it was difficult to replicate the original work without a certain amount of implicit knowledge from the original researchers. This involves either effort in sharing results or if this information is not shared, there will be badly replicated projects or non-combinable results. A good counter example has been the Reader's Project (Shull et al. 2002b), a collaboration between the University of Maryland and several Brazilian universities (Federal University of Rio de Janeiro, University of Sao Paulo, UNIFACS, UFSCar), where experiments were replicated, meta-analysis performed, and recognition of any problems identified. Joint funding was received to support this work, but such funding is limited and has since been terminated.

(3) *CeBASE—Center for Empirically Based Software Engineering: University of Maryland, Fraunhofer Center, Maryland and University of Southern California.* CeBASE was an NSF sponsored project to create a repository of "experience" associated with the application of various software development methods and techniques in order to create hypotheses, qualitative and quantitative models, and other forms of aggregated experiences (Shull et al. 2002a).

The original experience base maintenance was supported by the NSF grant, but once the grant ended, maintenance has become the responsibility of one of the CeBASE participants with the need to keep the repository alive. Similar to the NASA/GSFC SEL example, permanence of such data is a continuing issue.

(4) *HDPC—High Dependability Computer Program—University of Maryland, Fraunhofer Center, Maryland and University of Southern California.* This NASA-sponsored project was to define models of dependability and assess the ability of new research techniques to support the development of highly dependable systems for NASA (Donzelli and Basili 2006). In order to reduce the risk of applying these technologies to live systems before they are demonstrated effective, the project built experimental environments of testbeds to be used to compare the effectiveness of various techniques for improving software dependability in isolation and in combination. Because the project is no longer running, the following consequences and questions arose:

- A testbed, as a vehicle for evaluating various technologies, must continually evolve as new technologies are brought for evaluation. Who will pay for maintenance of the testbed, and if the testbed must be altered, who now owns it?
- How will the testbed and the results be maintained?
- How should the artifact creators be acknowledged?

(5) *HPCS—High Productivity Computing System Project—University of Maryland and Fraunhofer Center, Maryland.* This DARPA-sponsored program evaluates the workflows involved in developing programs for high end computers (HEC). Activities to date have focused on running experiments in parallel programming classes at seven universities in the USA in order to collect data on how graduate students develop such programs. The principal partners in the program have had to collect a core set of projects to give to students as well as develop data collection software and manual forms for students to fill out. But once the experimenters deal with student programs they have the legal issues of submitting applications to the various university Institutional Review Boards (IRB) for academic research on human subjects. Issues here, aside from the obvious ones of how do they maintain the data repository and allow for others to access it, are the additional constraints imposed by the IRB process—privacy issues of making sure no personal data about any participant is revealed and issues of student ownership of the actual programs they are developing (Hochstein et al. 2005).

(6) *Hackystat—University of Hawaii.* The Hackystat project explores automated collection and analysis of software engineering metrics. Sensors are attached to individual development tools, which unobtrusively collect data about product and process and send them to a central server where analyses of these data can be performed (Johnson et al. 2004). In addition to the 200 active users of the public Hackystat server, several groups have installed their own Hackystat server and are collecting their own data.

Hackystat-type data provides two immediate challenges: (1) *Privacy*: As in the HPCS project, what kind of data scrubbing process should be performed on the data in order to allow its publication without revealing identifying details? (2) *Context*: In direct contrast to the first challenge, without the addition of some demographic information about the developers, the product under development, and the context (process), it is unclear how the larger community could obtain value from this data. In summary, it appears that researchers must simultaneously add and subtract information from the Hackystat database to make it useful outside the projects in which the data was originally generated.

(7) *Sheffield Software Engineering Observatory—University of Sheffield.* The Observatory provides an environment in which student teams carry out various software

development projects for real clients (i.e. external to the university) under conditions that are as industrially realistic as is possible within a university context. Often several teams will work in competition, developing systems for the same client and requirements but using different development processes, which allows direct experimental comparison of these processes (Holcombe et al. 2003).

As with Hackystat, the two key issues surrounding the release of raw data are those of client and developer privacy. Some clients have required the researchers to sign non-disclosure agreements for the information about their businesses, and so the researchers in turn require all the students to sign such agreements. Thus, any release of raw data would be on the basis of the researchers to whom it was released signing a similar agreement, and this would have to require that it applied transitively to any situation in which they in turn were going to release some of the data. As far as the observational data is concerned this is dealt with partly by anonymizing it before it goes into the database, but in the project artifacts (such as plans and meeting records) and the quantitative data (such as timesheets) the individuals are often identified by name, and it would require considerable effort to anonymize them before release.

2.2 Why is Software Engineering Different?

One obvious question arises: Does empirical software engineering require different practices from other academic areas? The normal practice in many sciences is that in general experimental data and results are disseminated freely by publication, and once published are in the public domain (although still subject to the usual rules of copyright). Hence, the notion of ownership of data does not often arise; academic credit is gained from publication, possibly subject to conventions as to how credit is allocated between multiple authors. Other experimenters are in general free to do what they like with the data, although again this may be subject to conventions on how the originator of the data should be credited. But two assumptions made in most disciplines do not seem to apply to the software engineering domain:

(1) The context in which the data is obtained is sufficiently well understood so that all relevant parameters will form part of the published data or results to allow other experimenters to have all the information needed to use it. In the current state of empirical software engineering, though, it would be unrealistic to claim that the context of any study is sufficiently well understood to allow for the identification of the relevant parameters. Consequently, they certainly will not all be included in any publication of results. Thus it is inevitable that, as indicated above, other experimenters will need to interact with those who originated the data in order to be able to use it meaningfully. This increases the workload on sharing software engineering data from other disciplines.

The size of the data sets also affects this process. Software engineering projects are typically large, so the sample size under study is often small, with individual differences affecting the overall results. With medical or psychological studies, subject populations can be in the hundreds or greater, thus individual differences in subject behavior are minimized if these subjects are chosen randomly among the available population (e.g., the double-blind study).

(2) Scientific data is often impersonal and describes aspects of the natural world. This is not true of most software engineering projects; they are undertaken by people, for people, and hence (with the obvious exception of open source projects) they will nearly always have issues of personal or commercial confidentiality associated with them.

In other disciplines what is described as "experimental apparatus" covers a wide range of practice, as the possibilities vary from standardized pieces of commercially available equipment, such as measuring devices, to highly specialized or even unique machines, such as satellites, radio telescopes or particle accelerators. Across this range, though, there is a basic expectation that the design principles and parameters of such apparatus should normally be available publicly. Thus, in principle, experimenters should have a choice of buying some or all of this apparatus commercially, or of building their own, or collaborating with others who already have such apparatus, and in practice this choice will be an economic one rather than a scientific one.

Although this also applies to testbeds or artifacts in empirical software engineering, this assumption that the design principles and parameters that apply to any such apparatus is sufficiently well understood that other experimenters can build their own version and obtain the same results is not valid. Hence, the need to be able to achieve repeatability of experimental results in a much greater pressure towards collaboration in the sharing of such testbeds and artifacts than is the case in many other disciplines.

Furthermore, unlike studies in the social sciences, where individual people are subjects and represented as single data points within a large experimental set, for a software engineering project the specific characteristics of that one project may set the entire context for a whole study, meaning anonymity will be nearly impossible to achieve. Hence, empirical studies of real projects will usually need to be governed by some form of limited disclosure agreement, and any sharing of the data must make provision for these limitations on disclosure to be propagated.

3 Properties of Shared Data

Based upon the above experiences, the authors have identified the following properties related to the sharing of artifacts. As a minimum, the requester of the use of an artifact or data should (1) officially request *permission* to use the items (e.g., write a white paper on what they plan to do, ask permission), (2) *credit* the original developer with the work involved, and (3) provide *feedback* on the results of use as well as problems with using the artifact or data. (4) There are also issues concerning the *protection* of the data and artifacts like confidentiality and safety that need to be considered. For example, the data was most likely collected from people who were assured anonymity—how should this be handled? (5) The owner of the artifacts may be interested in the opportunity for *collaboration*. (6) Who handles *maintenance* of these artifacts?

The following explores the questions and some potential answers that each of these properties raise in more detail. A set of attributes are identified that may cross several properties, and an outline is presented of a classification system that can be used to define the rules relevant to sharing artifacts or data using these attributes.

1. *Permission*: Does one have to request permission to use the material? Is it simply publicly available? What should be the rules? If publicly available, how (or should) one provide some form of controlled access to the artifacts? Who owns the data? Is it

the authors (who created the data) or the organization the authors work for? There might be a request to use the artifact with a commitment to provide feedback after or during use (method, results, other data) and reference the items in all work using them. A mechanism that could effectively restrict access would be to require that the requestor write a short proposal to the data owner. Then the item can be used:

- Freely, in the public domain
- With a data sharing agreement or license
- With a service fee for use (by industry) to help maintain the data

The attributes of *lifetime* (time period artifact can be used), *area* (projects artifact can be used with), *transfer to a third party* and *derivative* (ownership of any derived artifact) are all related to the permissions property. Table 1 defines the set of proposed attributes.

2. *Credit*: How should the original group gathering the data or developing the artifact be given credit? What would be the rewards for the artifact or data owner? The type of credit is related to the amount of interaction. If there is an interaction, depending on the level, co-authorship may be of value. If it is used without the support of the data owner, some credit should still be given, e.g., acknowledge and reference the data owner. Thus, if the requestor uses it but the owner is not interested in working on the project, the minimal expectation is a reference and an acknowledgement. (There are various possibilities for how that reference should be made, e.g., the paper that first used the artifact, a citation provided by the artifact or data developer, or some independent item where the artifact itself exists as a reference.) It is also possible that some form of "associated" co-authorship might be appropriate. The attribute of *publication* addresses the credit property and the appropriate form of references to the artifact.

A side issue, independent of external interactions, is internal interactions. Experiments require a team of people. How is authorship of such work decided? Does everyone who contributes become an author? What should be the order of authors? Is there some form of guideline that should be provided here? For example, the University of Maryland has been criticized for having too many authors on a paper. This is common practice in experimental physics (everyone from the builder of the instrument to the original experimenter can be an author of a paper and author lists can extend into the hundreds), but it is not an accepted practice in computer science. For example, the author rating system used by one journal (*Journal of Systems and Software*) penalizes multiply-authored papers since the "value" of that publication is divided among all the authors.

3. *Collaboration*: In general it has been suggested that the requestor keep the option open of collaboration on the work. The Reader's project has been an excellent example of collaboration but required funding on both sides. Funding agencies are often looking for "new" ideas and so it is often difficult to be funded for a continuing operation. What options are there for funding collaborations? If collaboration is not desired by the owner of the artifacts, what are the rights and obligations of the requestor? The attribute of *Help* addresses collaboration issues.
4. *Feedback*: By requesting permission, there is a sense that the originators of the material know that someone is using their materials. However, feedback can act as a form of payment (e.g., updated versions of artifacts or data to allow its use in some form of meta-analysis, some indication of the effectiveness of technology in the experimental environment). A related issue is assuring that the quality of the data, analysis, and new knowledge being returned to the originator is acceptable and

Table 1 Data sharing agreement taxonomy

Attribute	Property	Option	Definition
Lifetime	Permission	Single use	Can use artifact only for one application
		Limited	Can use artifact repeatedly for a set period of time
		Unlimited	Unlimited use of the artifact
Area	Permission	Specific project	Can use artifact only for one or a specified number of projects
		Specific research	Can use artifact within one research area
		Unlimited	Unlimited use of the artifact
Data	Protection	Sanitized	No personal information contained
		Proprietary	Data contains information that uniquely identifies individuals of specific organizations
		Subset	Only a specified subset of the data may be used
Transfer to 3rd party	Permission	No	Only signer of agreement can use artifact
		Yes	Signer of agreement can pass on artifact under the same agreement conditions to another. This may require a non-disclosure agreement with the owner of artifact.
		Timed-release	Signer of agreement can pass on artifact after a period of time (e.g., restricted for 3 years, then available to anyone)
Publication	Credit, Feedback	None	Signer of agreement is free to use artifact in any way.
		Results only	Owner of data only wants results of using data or artifact.
		Prior results	Signer of agreement has to send results of using artifact to owner of artifact prior to writing a paper on the topic.
		Acknowledge	Signer of agreement has to acknowledge creator of artifact in publication. Agreement will state how this acknowledgement will occur. In some cases this may mean not to acknowledge creator of artifact.
		Review	Artifact owner has rights to review paper based on artifact prior to publication submission.
		Timed-release	Publication delayed for a specified time period (e.g., artifact has commercial value for a specified time).
Help	Collaboration	Data only	Signer of agreement obtains the data (and its metadata) or artifact "as is." No help is provided from artifact owner.
		Limited	Artifact owner is willing to provide limited help to signer of agreement to use artifact.
		Extensive	Artifact owner is willing to provide significant collaboration.
		Coauthor	Artifact owner and signer of agreement agree to coauthor publication.
Costs	Maintenance	None	Artifact is free to signer of agreement, with perhaps a minimal cost for a tape or CD of data
		Payment	A set amount (as money or "in-kind" contribution) is specified to obtain artifact. This may help provide funding for maintenance of artifact repository.

Table 1 (continued)

Attribute	Property	Option	Definition
Derivatives	Permission, feedback, protection, maintenance	None	Derived artifact is owned by signer of agreement. (May be separate clauses covering derived software and related artifacts or derived data using meta-analysis)
		Evolution	Artifact creator must receive derived artifact. Ownership of artifact may be creator, signer of agreement, or both jointly.
		Enlargement	A new artifact or data is added to data set. Ownership of artifact may be creator, signer of agreement, or both jointly.
		Open-source	An agreement such as used by the open source community from the Free Software Foundation. Any derived work has the same usage requirements as the original artifact.

consistent within the context of the original experiment. The previously mentioned attributes of *publication* and *derivatives* also affect feedback.

5. *Protection*: There are a large number of issues here. How does one limit potential misuse? How does one support potential aggregation and assure it is a valid aggregation. How does one deal with proprietary data? What about confidentiality? The *data* as well as the *derivatives* attribute affects the protection property.

What is required of the originators? Should they be allowed to review results before a paper is submitted for external publication? Do the artifact owners have any rights to stop publication of a paper with invalid results based upon the original artifacts or is the "marketplace of ideas" open to badly written papers? Should there be some form of permission required by reviewers? Who has the rights to analyse and synthesize and create new knowledge based upon the combined results of multiple studies? Again here, how is credit given or authorship determined? How does one limit potential misuse?

On the other hand, how do we protect scientific integrity? Readers of peer-reviewed papers should have a right to analyse the data in the paper to see if it is correct. How? If users of data find gross negligence on the part of those who created it, what are their obligations to reveal those issues (e.g., the South Korean scandal over stem cell research[1])? Can data sharing requirements be an impediment imposed by the guilty to hide their actions?

6. *Maintenance*: A large physical device (e.g., particle accelerator) generally is built and supported over the long term. But the same has not been true of computer software, which has an ethereal quality of simple residing hidden in a computer file system. Who pays for the cost of maintaining the experience base? There are only three possibilities here toward maintenance: (1) Owner of the data, (2) Users via a usage fee, (3) Everyone via an open source arrangement. "Owner of the data" generally won't work since few have such resources, "Usage fee" may work, but costs will limit use; researchers won't generally pay for something they view as a "free resource." "Open source" is a possibility. The *costs* as well as the *derivatives* attribute affect the maintenance property.

[1] South Korea: Scientist Admits Faking Stem Cell Data, New York Times, July 5, 2006.

The open source community has figured out an economic model that seems to work. Systems such as the Apache web server, Linux kernel, the Eclipse environment are all open source and freely available to anyone for download. However, companies have made valuable additions to these systems by adding proprietary additions to the basic system or by providing maintenance of them, and thus generate a revenue stream to keep the open source movement viable. The same model, to date, does not seem to work with data. Researchers, the primary users of such data, do not have the resources to create the economic demand for this data.

4 Attributes of a Data Sharing Agreement

The previous section has developed a proposed set of attributes that can form the basis of a data sharing agreement for empirical software engineering artifacts. Table 1 collects and defines the attributes with their options and indicates which of the above properties they address. From these attributes, a software agreement, using one or more options for each attribute, can be written for any artifact, which defines the protocols required by both developer and user of that artifact. In the Appendix a sample agreement is given for the NASA SEL project, described previously, to show how this taxonomy can be used.

If we can establish a website for collecting and publishing details about organizations who sign such artifact sharing agreements, we can create a culture where only data whose provenance is traceable through this website will be readily accepted by the community. This will foster additional adherence to these guidelines and encourage others to participate.

We also have to point out two important issues that were not part of this study: (1) We do not provide for the quality of the various artifacts. We believe that is outside of the realm of a sharing agreement and a user of someone else's data needs to be sure the data is of the quality desired. (2) Providing a set of artifacts has intellectual property issues (including the value of that intellectual property) since the creator of the artifact has invested knowledge in creating the artifacts and data sets. This needs to be considered when the artifact creator and user of the artifact define a sharing agreement.

5 Conclusion

This paper has tried to define a structure to allow for the sharing of empirical software engineering artifacts, including data and testbeds. While this structure may not have achieved the ultimate set of properties necessary for an artifact-sharing agreement, the authors believe that it provides a good basis that should help foster empirical software engineering research using a set of universally available artifacts. If you, the reader, have a data set or know of one, can you specify its use via the attributes of Table 1? If so, let us know your experiences with it.[2] The hope is that this paper will stimulate discussion within the research community, so that a more comprehensive protocol for using artifacts and empirical data can eventually be established within this community.

Acknowledgement This paper is an outgrowth of sessions at the International Software Engineering Research Network (ISERN[3]) meeting in Los Angeles (2004), Noosa Heads, Australia (2005) and Rio de Janeiro (2006). ISERN is a worldwide organization of about 50 organizations dedicated to fostering empirical software engineering research. The authors acknowledge the contributions of the 75 attendees to those meetings.

[2] Please send comments to the second author, Zelkowitz.

[3] http://www.iese.fhg.de/isern/.

Appendix

Potential data sharing agreement for NASA SEL data

The attributes that most closely reflect what was done with the NASA SEL data in the 1990s are given below.

Attribute	Option	Sample agreement wording
Lifetime	Unlimited	Signer of the agreement may use artifact as long as needed.
Area	Unlimited	Signer of the agreement may use artifact on any project
Data	Proprietary	Artifacts contain information that uniquely identifies specific projects and personnel. Any report developed from this data must remove all such personal identifications.
Transfer	Yes	Signer of the agreement is free to transfer data to any other organization.
Publication	None	Signer of agreement is free to publish all results using these artifacts
Help	Limited	Owner of data will give limited help in using this data.
Costs	None	There is no cost to obtain a tape of this data.
Derivatives	None	Any derivative work is owned by the signer of the agreement.

If those responsible for this project were writing such an agreement today, in the light of the experience described above it probably would be more restrictive, as follows:

Attribute	Option	Sample agreement wording
Publication	Acknowledge	Signer of agreement must acknowledge the owner of data as the source of the NASA SEL data in any resulting publications. (Perhaps also a clause that signer will give owner prior results to allow for checking whether data is misused.)
Derivatives	Open-source	Any derivative work is covered by the same conditions as this agreement. A copy of the derived result shall be transmitted to the owner of data.

References

Basili V, McGarry F, Pajerski R, Zelkowitz M (2002) Lessons learned from 25 years of process improvement: the rise and fall of the NASA Software Engineering Laboratory. IEEE computer society and ACM international conference on software engineering, Orlando, FL (May)

Donzelli P, Basili V (2006) A practical framework for eliciting and modeling system dependability requirements: experience from the NASA high dependability computing project. J Syst Softw 79 (1):107–119 (January)

Hochstein L, Carver J, Shull F, Asgari S, Basili V, Hollingsworth JK, Zelkowitz M (2005) HPC programmer productivity: a case study of novice HPC programmers, supercomputing 2005, Seattle, WA (November)

Holcombe M, Cowling AJ, Macias F (2003) Towards an agile approach to empirical software engineering, proceedings of the workshop on the future of empirical studies in software engineering, Rome, Italy, Fraunhofer IRB Verlag, pp 35–46

Johnson PM, Kou H, Agustin JM, Zhang Q, Kagawa A, Yamashita T (2004) Practical automated process and product metric collection and analysis in a classroom setting, International symposium on empirical software engineering, Los Angeles CA (August)

Shull F, Basili V, Boehm B, Brown AW, Costa P, Lindvall M, Port D, Rus I, Tesoriero R, Zelkowitz M (2002a) What we have learned about fighting defects, IEEE computer society international symposium on software metrics, Ottawa Canada, pp 249–258 (June)

Shull F, Basili, V, Carver J, Maldonado J, Travassos G, Mendonca M, Fabbri S (2002b) Replicating software engineering experiments: addressing the tacit knowledge problem, International symposium on empirical software engineering, Nara, Japan (October)

Victor Basili is a professor of Computer Science at the University of Maryland. He holds a Ph.D. in Computer Science from the University of Texas and honorary degrees from the Universities of Sannio (Italy) and Kaiserslautern (Germany). He was a founding director of the Fraunhofer Center - Maryland and the Software Engineering Laboratory (SEL) at NASA/GSFC. He works on measuring, evaluating, and improving the software process and product. He has authored over 200 journals and refereed conference papers. He is a recipient of the 2000 ACM SIGSOFT Outstanding Research Award, 2003 IEEE Computer Society 2003 Harlan Mills Award and awards from NASA and the Washington Academy of Sciences. He is an IEEE and ACM Fellow.

Marvin Zelkowitz is a professor of Computer Science at the University of Maryland and Chief Scientist at the Fraunhofer Center, Maryland. He has a BS in Mathematics from Rensselaer Polytechnic Institute, an MS and Ph.D. in Computer Science from Cornell University and is a Fellow of the IEEE. He was one of the directors of the NASA Software Engineering Laboratory (1976–2002) and has studied software development for 35 years. His research interests include empirical software engineering and technology transfer with current interests in computer security and the software engineering of high performance computers.

Dag I.K. Sjøberg received the MSc degree in Computer Science from the University of Oslo in 1987 and the Ph.D. degree in Computer Science from the University of Glasgow in 1993. He has 5 years of industry experience as a consultant and group leader. He is now research director of the Department of Software Engineering, Simula Research Laboratory, and a professor of software engineering in the Department of Informatics, University of Oslo. Among his research interests are research methods in empirical software engineering, software processes, software process improvement, software effort estimation, and object-oriented analysis and design.

Philip M. Johnson is a professor of Information and Computer Sciences at the University of Hawaii and Director of the Collaborative Software Development Laboratory. He received B.S. degrees in both Biology and Computer Science from the University of Michigan in 1980, and M.S. and Ph.D. degrees in Computer Science from the University of Massachusetts in 1990. He has published over 50 papers in areas including software engineering, computer supported cooperative work, and artificial intelligence.

Tony Cowling is a member of the Verification and Testing research group in the Department of Computer Science at the University of Sheffield. He has been a lecturer there since 1973, having previously obtained both his BSc and his Ph.D. from the University of Leeds. He is now a senior lecturer and the director of teaching quality for the department. Professionally, he is a member of the British Computer Society, a Chartered Engineer in the UK, and a European Engineer (Eur Ing). His research interests are in the areas of software engineering education, the formal modelling of software systems, software testing and empirical software engineering, and he has published over 30 journal papers in these areas. Currently, he is one of the investigators in the research team for the Sheffield Software Engineering Observatory project (referred to in the paper), and as part of this, he represents the Verification and Testing research group in the ISERN network.