SPIQ: A Revised Agenda for Software Process Support

Reidar Conradi
Norwegian University of Science and Technology (NTNU),
N-7034 Trondheim, Norway
Phone +47 73.593444, Fax: +47 73.594466, Email conradi@idt.unit.no

1 Introduction

In the last decade, there has been much interest in the software process as a vehicle to improve software production. Many Process-Centered Software Engineering Environments (PSEEs) have been constructed and documented, although to a less extent tested in industrial settings. It is now time to assess progress and set course for future efforts. That is, what is now ready for deployment, where is future effort needed, and where do we have to revise goals and strategies?

The paper is written as a precursor for a Norwegian industrial SPI program, called SPIQ or Software Process Improvement for better Quality.

2 Approaches to Software Process Improvement (SPI)

We can divide the SPI support camp in six categories:

- **The quality certifiers**: Using ISO-9001 [Inc94], TickIT, CMM [PWCC95], SPICE [Dor93], BOOTSTRAP [HMK’94] etc. to improve an organisation’s formal process standing. The underlying assumption is that better process(-indicators) will lead to a better product, predictably on time and budget. However, there are no validated studies to confirm that an ISO-9001 certification or a CMM-level n+1 classification represent a true SPI capability. However, most SPI meta-methods (QIP, CMM, ISO-9000) are very general, and lack immediate usability. Further, most SPI technology are not geared towards small development teams or companies, possibly operating in a very changing business environment. E.g. a recent CMM study include 13 organisations with an average size of 200 developers [HCR’94], and with very stable project models.

- **The formalists**: Better process modelling languages (PMLs), as exemplified by European SPI research in the first PROMOTER book [FKN94]. However, there is no agreement on the core process elements [CL95] (activities, products, tools, humans/roles, evolution etc.), or the linguistic paradigm to be used. Most software developers also feel entranged by current PMLs, conceptually and syntactically. What really counts is simplicity, modularity, standardisation, user-friendliness (e.g. good graphics), and easy evolution of the process model. A possible path is therefore to design more high-level and
user-oriented PMLs, and to translate these down to simpler ones, that can be supported by commercial PSEE or workflow tools, see next point.

- **The tool advocates:** Computer-assisted SPI support by enactable models in a PSEE or a workflow tool [Fer93] [RP95]. Such support invokes endless discussions on the level and strictness of machine intervention. ie. should the emphasis be on assistance, guidance, enforcement, automation, monitoring etc.? A good user interface [Mye89] and tool interoperability [ECM91] are also crucial. This leads us into interoperability issues on how newer tools for workflow, CSCW, configuration management, databases (CORBA) etc. can cooperate on a distributed and federated architecture, using industry-standard PCs and not Unix workstations.

- **The practitioners:** Emphasising mainstream project-, quality- and product management, applying project-level planning and metrics, but with little formal process modelling or enactment support. This is what most software teams are striving for today. We can mention the work by NASA-SEL, resulting in the Quality Improvement Paradigm (QIP) as part of the Software Experience Factory [Bas93]. There is also the AMI method to establish a metrics [AMI95]. However, few PMLs and PSEEs can offer project management to support or utilise a metrics database.

- **The organisation reformers:** The goal is an understood and possibly restructured processes in a learning organisation, often involving an experience database – cf. point above. This involves human and organisational issues (so-called humanics), and may involve Software Engineering Process Groups (SEPGs). That is, how are organisational needs formulated, and which of these can and should be covered by SPI, and how? Many SPI programs have failed because of organisational inhibitors, so this is a crucial point.

- **The ideologists:** Better understanding of SPI paradigms and concepts, cf. fundamental work by Manny Lehman [LB85] and terminology work in [CFF94]. For instance, which categories of processes are most suited for formal SPI support? – see also points on practitioners and organisation reformers above. Many software teams claim that the most difficult and risky process phases are customer negotiation and project planning, not “run-of-the-mill development” emphasising reliability (QA). Likewise, the most interesting processes for software reuse are not so easy to support, namely reuse management across divisions and product lines, and design of “pluggable” software architectures. Lastly, we must arbitrate between “armour or mobility” (after von Clausewitz, mentioned by [DeM96]), i.e. controlled stability or manouevrability to chase evolving markets.

Further, there are many related disciplines [TC95], e.g. information engineering and conceptual modelling, enterprise modelling and business process reengineering, workflow and CSCW, tool architectures for distributed / federated systems, configuration management, project and quality management etc. etc. Thus, what belongs to the core software process field, and what can more easily be fetched from other disciplines? For instance, there is a 2 bill. USD workflow industry.
This implies, that formalists and tool advocates should advice software developers to use commercial workflow tools, both for process modelling and enactment.

3 The road ahead

The ultimate goal for process research should be to achieve demonstrable SPI among software practitioners and to install a SPI culture in their organisations, so that the practitioners here and elsewhere can improve and learn from practical experiences. Most PMLs and PSEEs have been an “oversell” in relation to the perceived SPI needs and priorities – with marginal impact on practice, except from a possible increase of process awareness. This will force university researchers over to the “method side” in their theoretical work.

That is, the universities should advice on how to manage SPI introduction and related experiences, rather than develop own languages and prototype tools. In short: they should actively apply and add value to commercially available SPI technologies, thus joining researchers, vendors and practitioners.

On the other hand, SPI researchers need to formulate and validate theories, methods and hypotheses for real software production. For a university community, this involves a delicate balance between mainstream “consulting” to get any SPI technology into use, and doing publishable, theoretical work that can be validated in practice. This theory/practice dilemma is especially pressing for software engineering and SPI, since most SPI researchers at universities have little background in technology transfer and organisational sciences. SPI technology is not like a compiler, that can be tested exhaustively by the developers themselves.

However, the theory/practice gap is ameliorated by a continuum of experimental SPI opportunities, ranging from student projects and laboratory experiments via pilot projects to full-scale software projects in industry. Thus, researchers should provide to practitioners new or revised methods and hypotheses, possibly with some languages and instrumented tools. These hypotheses should be tried out in an industrial setting, either real or simulated. Inversely, industry should return real-world and metricated process models and experiences with these. This feedback can be analyzed in a research situation, and provide ground for revised SPI technologies, and so on. Such a mutual and long-term cooperation on SPI issues is often lacking.

Some concrete recommendations, organised along the previous categories:

The quality certifiers:

– Identifying a general SPI framework. As a start, we could apply e.g. the following meta-activities from QIP: S1) Establish a process status; S2) Formulate quantitative SPI goals; S3) Make a metricated and revised SPI plan (a process model); S4) Perform ensuing software development with measures and with plan adjustments; S5) Analyze the results; S6) Package and learn. Steps S1 and S2 will have an immediate value regardless of the SPI context. However, the application of steps S1-S6 may need extra expertise and experience, as QIP is too “high level”. We may need to narrow the SPI focus
in status/goals/planning (S1-S3). We should also consider downscaling SPI frameworks, e.g. a diet-SPICE or mini-BOOTSTRAP (the latter for S1).

- **Validating SPI paradigms such as QIP and CMM**, challenging their “assembly-line” views of human processes and process evolution (S1-S6).

- **Formulation of risk and cost/benefit models** to compromise between conflicting goals, especially for smaller companies (S3, S6).

The formalists:

- **Exploring the role and range of formal process modelling**, e.g. what PMLs are most appropriate for what processes, and in what SPI phases (S2-S4). Formal PMLs could be used for definition and analysis of models, but enactment support may be partly manual, or hybrid.

The tool advocates:

- **Applying and adding value to commercial PSEE/workflow tools**, and gradually exploring new platforms like Java, Perl and Tcl/tk (S3-S4). A range of PMLs, from rather informal prose/diagrammatic formalisms, to more high-level ones could be used, depending on circumstances. We should also analyze the coupling between configuration management, CSCW and process support, with emphasis on interoperable architectures (S3-S4).

The practitioners:

- **Identifying incremental and concrete results**, i.e. to have some initial results after 1 year (using existing technologies), some more consolidated ones after 2-3 years (adapting existing technologies), and some more sophisticated results after 4-5 years (partly using new technologies). We should, however, avoid technological oversell and promises of fast and spectacular SPI gains (S1-S6).

- **Reusable processware** (process models, quality models etc.) must be synthesised, facilitating customisation as well as evolution of process models, cf. point below (S2-S6). This is the ultimate vehicle to achieve long-term SPI!

- **Experience database**, one per company, should be exploited and disseminated, e.g. in ACCESS-DBMS and exploiting WWW (S1-S6). However, much experience information is also qualitative.

- **Experience networks** should be established, e.g. in the form of national SPINs (S1, S6).

The organisation reformers:

- **Long-term management support** must be solicited and secured.

- **Humanics** (human, social) issues, e.g. establishing SEPGs to pioneer SPI in the short term, and analyzing why SPI and metrics programs fail in the long term (S1-S6).

- **Empirical studies of software developers** should be performed, an often neglected field (S4).
The ideologists:

- **Identifying the critical processes.** For instance, more emphasis might be put on software (sub)processes such as project preplanning, use of subcontractors, process evolution and studying general “disturbances” (S4).
- **Interdisciplinary teams** are needed, with competence especially in SPI, quality management and project management, but also organisational sciences, technology transfer, and statistics (S1-S6).

4 One Possible Research Context: The SPIQ Program

As mentioned, SPIQ is sought to be executed in 1996-2001 by four research and 10-12 software producing organisations. The latter are mainly small and medium enterprises. They operate in different domains, apply different development technologies, and represent different sizes (teams of 2-10 persons) and company cultures. The SPI emphasis will be on different development phases (planning vs. requirements vs. testing), and be driven by different needs (time, budget, reliability etc.). This heterogeneity is both a problem and an advantage.

The industrial goal is to realise a 30% SPI saving in costs in these companies within 5 years, and to convert these companies into learning organisations. The research goal is to produce a Norwegian SPI method book by downscaling, adapting and enhancing existing SPI methods. An experience database will be exploited, possibly augmented by commercial workflow/PSEE tools and appropriate high-level PMLs. SPIQ is planned with three main phases over 5.5 years:

**Phase 1** (1.5 years): Getting started and running the first SPI experiments. An initial SPI method book for Norwegian SMEs and a demonstrator experience database will be provided.

**Phase 2** (two years): Refining the experiments on similar processes. A complete method book, experience database, and high-level PML will be provided.

**Phase 3** (two years): Consolidating the experiments on other kinds of processes. A final method book and experience database will be provided.

5 Conclusion

SPI researchers and practitioners must find each other, so that progress on SPI technology can go hand in hand with demonstrable and long-term SPI results. The Norwegian SPIQ program will define a joint work place for such efforts.

References


