Requirements on software process technology from the viewpoint of commercial software development

Recommendations for research directions

Ralf Kneuper

TLC Transport-, Informatik- und Logistik-Consulting, Frankfurt/M, Germany
ralf.kneuper@gmx.de

Abstract. One of the main users of software process technology (SPT) are Software Engineering Process Group (SEPG) responsible for the development and implementation of a company's software process model (SPM). This paper discusses some research directions in SPT research (cf. [5]) from the perspective of a SEPG. In the form of requirements on SPT, the paper points out research directions that will solve genuine problems in the work of an SEPG, and others that will only be nice-to-have or even irrelevant.[1]

1 Requirements

The main application areas of software process technology (SPT) are company software process models (SPM) and public standards. The emphasis here will be on company SPMs.

Note that in this paper, only the technical aspects of SPMs are discussed. It should be kept in mind that the success of an SPM depends more on the organisation and the processes supporting the SPM, cf. [3].

Requirement 1 *In the context of commercial software development, we need prescriptive SPMs that guide and support the developer. Being able to* analyse *the SPM in detail (consistency checking, etc.) is useful but clearly second priority.*

Descriptive modelling is used for analysing the process by consistency checking, static analysis and dynamic analysis, while prescriptive modelling is used in development for

- guiding the user, i.e. developer
- supporting developers by automating certain tasks

[1] The author has been involved in developing, promoting and implementing company software process models in several different environments and is currently responsible for quality management and software process modelling in a large application development project.

In a commercial environment, process models are only relevant if the process modelled is actually being followed. To be followed, the model must be prescriptive and provide at least strong guidance to the developer. Consistency checking of the SPM, on the other hand, is not a serious problem in this environment. If an SPM is really used, then any major inconsistencies are discovered very quickly and only minor inconsistencies can slip through, which can easily be corrected by the developers.

The latter is an important aspect of software processes that e.g. Osterweil in [4] missed by stressing only the programming and automation aspect of process modelling. Software development is done by people, who can adapt to minor inconsistencies and repair them and who need a different kind of support for their work than machines.

Nevertheless, being able to check certain consistency conditions automatically would certainly be useful and reduce effort as long as it does not interfere with the usage of the SPM by people, as discussed in the following requirement.

Requirement 2 *We need methods and tools to create software process models which are easy to understand and easy to apply*

This really follows from Req. 1 since it is mainly important for prescriptive models to be used by developers. Developers will only accept a model if they can easily use it and understand it, while a small group of process engineers (such as the SEPG) can be expected to deal with a more complex notation, as is necessary for descriptive models used for analysing processes.[2]

If SPMs are to be easy to use, most formal methods/notations are not immediately appropriate, since formal models are usually not readable for developers. On the other hand, in order to apply analytic approaches for consistency checking etc., a certain degree of formality is needed. Ideally, we would therefore like to have a notation that is both (semi-) formal and easy to understand.

Requirement 3 *We need methods and tools to create software process models that are well-structured*

This requirement is closely connected to the previous one, since a clear structure helps to make an SPM easy to use and understand.

Currently, the different SPMs all have very different structures. If a company intends to develop its own SPM, there is no obvious, standard structure for it to use. Furthermore, it is difficult to compare different models since the same content might be packaged very differently.

To overcome these difficulties, a meta-model that describes a general structure of SPMs would be helpful. This structure must cover both life-cycle (coarse-grain) and development (fine-grain) models, preferably also process engineering processes such as improvement or reuse processes. It has to cover not just a

[2] Note however that in many cases an SPM will be the joint work of process engineers in the SEPG, and developers who join in the development of the SPM as a minor part of their job

single process but the collection of all software processes relevant to the organisation (SPM repository) and it should provide enough formal structure to allow automated analysis such as consistency checking of the resulting SPMs.

Requirement 4 *The SPM used by the developer must be the same as or at least automatically derived from the SPM used for consistency checking, static analysis, etc.*

To make an SPM usable by people *and* perform automated consistency checks, one approach is to create two versions of the SPM, one in a human-readable form, the other in a formal language suitable for automated analysis. For example, in [6] Verlage describes the formalization of existing informal SPMs in order to check certain consistency conditions.

In this case, consistency between the two versions of the SPMs is needed, since otherwise internal consistency of the formal SPM is worthless. To achieve consistency, preferably one should deal with only one version of the SPM (which would have to be the formal one), and derive the other (easy-to-understand one) automatically from it.

This implies that the formal version of the model would need to have some unstructured, informal components (e.g. text fields for describing manual steps in the model).

If, on the other hand, the process model used by developers is derived *manually* from the formal model, then

- either the model is static and is only updated rarely. This is appropriate for process models with a large scope, such as a national standard. In this case, a formal model can be very useful [6] since inconsistencies affect a large number of people and are difficult to correct once found, while on the other hand keeping the models consistent is not such a large effort.
- or the model is adapted and improved in short intervals (continuous improvement process). For company process models, this is usually much more appropriate, but then the formal model and the model used by developers will diverge fairly soon, since it seems unrealistic to expect that one will be able to maintain consistency between the models manually without excessive effort.

Requirement 5 *We need methods and tool support to move from a general SPM to a project-specific SPM and on to a project plan (tailoring)*

First of all, tailoring is concerned with questions such as "When do you add or delete a certain task?" and the follow-up question of who is allowed to decide how to tailor an SPM in any given context. Although in principle, tailoring is not difficult to describe, it soon gets quite difficult to actually do it due to the large number of dependencies that need to be tracked.

This leads to the obvious question of tool support for tailoring, and there are a number of tools that support tailoring to some extent (see e.g. [1, App. A] or [2]). However, in order to fully support tailoring, tools need to satisfy the following:

- support for maintenance of differently tailored versions of a process model, in particular if continuous improvement is wanted (see Req. 7) and the SPM therefore changes continually. Usually, in tailoring an SPM an intermediate step of creating different project types with some common features is used.
- coverage of the whole tailoring process including the final step of deriving a project plan. Since a project planning tool will usually already be available, this requirement will have to be implemented as an interface from the SPM/tailoring tool to the project planning tool.
- the tailoring tool (including the interface just mentioned) has to be very flexible; e.g. during functional design, it is often not yet known how many modules will be needed [5].

Requirement 6 *We need tool support to create SPMs and publish them both on paper and in an electronic format. The electronic format must be accessible from different technical environments*

While the need for tool support to *create* SPMs is fairly obvious, the need to publish the result to developers is less widely acknowledged in work on SPT. However, without that, developers will not even be able to use the SPM. Note that the format of the published SPM needs not be identical to the format that is used for development of the SPM, but it must at least be automatically derived from it (similar to Req. 4).

A paper version in addition to an electronic version is e.g. needed in order to hand out (parts of) the SPM to external partners, such as subcontractors or customers. Occasionally, the paper version will even become part of a legal contract.

The electronic version, on the other hand, must fit into different technical environments since no company will be prepared to change their technical infrastructure just to be able to access an exotic format of the SPM.[3] Typical formats that are accessible from a wide range of technical environments are Acrobat PDF, HTML, or, to a lesser extent, Lotus Notes. Less suitable formats in most environments are any new tools, especially if they are academic prototypes not available as fully supported products, or CASE tools.[4]

Earlier, the need for a meta-model (Req. 3) for SPMs was stated. Note that in order to apply this, the meta-model must be broken down to the level of the tools used for creating the SPM, answering questions like "How do you map the object 'activity' in the meta-model to an object in Acrobat PDF?".

Requirement 7 *We need support for continuous improvement of SPMs*

[3] Note that this applies to publication of the SPM, not its creation. Having to deal with a tool that does not fit into the technical infrastructure is not usually a serious problem as long as only a few people, such as the SEPG, are affected.

[4] A CASE tool as medium for publishing an SPM tends to be suitable if and as long as it is also used for development. However, large companies rarely manage to restrict themselves to only one such tool, and even if they do they will sooner or later want to move to a different tool, or at least move up to a new, not-quite-compatible version.

Process improvement of SPMs is widely acknowledged as a tool to keep a software development organisation competitive, and the possibility to use explicitly defined SPMs as the starting point for process improvement is one of the major arguments used in favour of them.

To put this into practice, an immediate feedback mechanism is recommended, e.g. by adding a feedback button into the SPM. However, a number of unsolved difficulties remain:

- How do you support continuous improvement, in particular while a project is running? What if the project has already tailored the general SPM to its specific needs? To handle this, SPT need to support several versions of an SPM in parallel, where some improvements/modifications are applied to all versions, others to only some of them.
- How often do you update an SPM assuming that the tool/infrastructure supports it? The question here is whether to provide genuine continuous improvement or whether to version the SPM. Again, this is a standard question in change management that needs to be answered in the context of SPMs.

2 Summary

This paper describes the needs of commercial software development regarding SPT. Commercial software development needs SPMs that guide and support the developer in their task. In order to achieve that, SPMs have to be easy to use and well-structured. Furthermore, they have to support tailoring and publication of the SPM in both paper and electronic versions. Last but not least, the SPM should not be static but subject to a continuous improvement process.

References

1. F.C. Budlong, P.A. Szulewski, R.J. Ganska: Process Tailoring for Software Project Plans. Software Technology Support Center (STSC) (1996)
2. H.-L. Hausen, D. Welzel: Tailoring and Testing of Process Models Applied to the V-Model. In: S. Montenegro, R. Kneuper, G. Müller-Luschnat (eds.): Vorgehensmodelle — Einführung, betrieblicher Einsatz, Werkzeug-Unterstützung und Migration. GMD-Forschungszentrum Informationstechnik, GMD-Studien Nr. 311 (1997) 7–13
3. R. Kneuper: Organisatorische Gestaltung des Einsatzes von Vorgehensmodellen. In: R. Kneuper, G. Müller-Luschnat, A. Oberweis (eds.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung Verlag B.G. Teubner (1998) 228–248
4. L. Osterweil: Software processes are software too. In: Proceedings of the Ninth International Conference on Software Engineering (1987) 2–13
5. H.D. Rombach, M. Verlage: Directions in Software Process Research. Advances in Computers **41** (1995) 1–63
6. M. Verlage: Vorgehensmodelle und ihre Formalisierung. In: R. Kneuper, G. Müller-Luschnat, A. Oberweis (eds.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung Verlag B.G. Teubner (1998) 60–75

This article was processed using the LaTeX macro package with LLNCS style