On Formalizing Software Process Description: Consortium’s Example

Robert Chi Tau Lai
Software Productivity Consortium
lair@software.org
(703) 742-7126

1 Introduction

It is useful to try to formalize the description of the software process so that one may understand it better; so that one may provide automated support for it; and so that one may provide guidance to software developers. But, software developers are human beings. They are familiar with the natural language and will be more responsive to the process description in the language they know well. This paper describe the three different tools of formality on process description for human (process guidance) and machine (automated process support). Three level of process descriptions suggested by this paper are: process guide book, process model and process notation. The natural and direct mapping between them will free the software developer to respond to the process description in the language they know well. This paper describes the three different tools of formality on process description for human (process guidance) and machine (automated process support).

When a software development process is in formal description, 'formal' here only means that everything is based on pre-defined terms, the reasoning about various process management problems can be carried out. Some of the examples from the existing published papers are automated process enaction, process management, process control, process simulation, process design, process planning, as well as the automation of matrix data collection [8]. One more benefit is the support of automated generation of the pointer or heuristics for possible process improvement. Recently, more and more literature already reported the transition of software process modeling techniques into practical use.

2 Software Process Guidebook

Software Process Guidebook is, SPG, the material/document that every software developer will have. SPG represents the software development procedures for an organization/division/team. The formality in the SPG will start with the unified and consistent (1) term definitions, (2) section formats, (3) decision tables/forms, (4) types of diagram to represent activities, and (5) software process and product matrices collection tables. The main purpose of formality for SPG are (1) software developer can reason, reference, argue and then follow the description to work and record the necessary data, and (2) the formality of the SPG, formal definition of types of form/table, can be used to check the completeness, consistency, conflicts and reachability of the guide book. Two examples for the SPG are (1) Software productivity Consortium’s "Evolutionary Spiral Process Guidebook," [13] and (2) STARS’s "Cleanroom Engineering Software Development Process Guidebook." Three types good example diagram for SPG are (1) SADT[9], (2) ET/UX [11] and (3) formal block structure diagram [5].

3 Software Process Model

A Software Process Model, SPM, is an essential foundation for the connection and mapping between the SPG and software process notation. The formality inside the SPG implies the SPM. SPM is used both as conceptual model for the software developer to think and as internal model for the implementation of checker for completeness, consistency, and reachability of SPG. A SPM offers a SPG with (1) possible relations among different kinds of artifacts in software process, (2) possible sequence of activities in a process, (3) possible partition of software products, and (4) a set of possible activities. SPM encapsulates a family of processes which can be instantiated from the model. Two of the most famous process model are Waterfall Model [12] and Spiral Model [1]. Other example of SPM are Rome Air Development Center’s Knowledge-Based Software Assistance, KBSA, process model [4], Software Productivity Consortium’s Synthesis Process [2], 2167A [3], and IEEE life cycle standard [6]. A software organization may have its own software process model which reflects its software development experiences, procedures, policies and conventions.

4 Formal Software Process Notation

A Formal Software Process Notation is capable of representing different kinds of SPM. It represents difference elements of the process model formally. There should exist a mapping from the FSPN to software development environment. So that a process represented in FSPN can be interpreted and software process management environment can be generated. Some examples for FSPN are APPL/A [14], and STATEMATE [7] and Software Productivity’s two-level state model notation. Software processes written in FSPN can be analyzed, simulated and automated. The implementation of FSPN can automate the data collection for software process and product measurement.

5 Software Development Equipped With Formal Model

One possible scenario of a software organization using all of these three together will be:

1. to analyze their existing process by writing SPG,
2. to summarize the pieces of process to form SPM which represent the current process and the future process,
3. to modify the SPG base on SPM,
4. to use a FSPN to represent the SPG,
5. to analyze the SPG represented in FSPN,
6. to use the SPG for software projects with all of the data recording tables/forms,
7. to analyze the data collected and decide the possible area of process improvement and then
8. to make change on SPG and SPM as necessary and go to step 4.

6 Consortium's Example

The Consortium is undertaking the task of software process improvement via (1) conducting Software Engineering Institute process self-assessment for member companies, (2) developing a guidebook for practise of the spiral process, (3) prototyping an environment based process management tool for supporting automated software development. Consortium member companies will be able to use this process notation to (1) plan future development processes, (2) understand and control current processes, (3) plan process improvements for the future, (4) serve as basis for automating of process management functions, (5) encapsulate software process fragments as reusable assets, and (6) govern the subcontractors by agreed and predefined process through process control described in Consortium's formal process notation.

A company may use this process notation to model a class of processes which encapsulate the different possible processes which implements the company’s policy, procedure or a process life cycle standard. One example of this is Consortium’s attempt to model a class of evolutionary spiral processes. The second kind of usage of this notation described in (2) above is for a line manager to model the process for his project. A tool developed to support notation would also be able to answer questions such as: What is the difference between version X and version Y of the software? What modifications have been made since a design artifact reached a particular state, or since a particular P-state was last visited? What modules are affected by this or that particular modification or batches of changes or modifications? Who modified them, and for what reason? What remains to be done to complete a particular state? What are the states of various design artifacts?

7 Conclusion

Consortium is extending its formal process notation in three directions, first, to strengthen the process model’s mathematical foundation, second to add organizational model to represent the structure and relations among software professionals in a software process, third to add a resource model for representing the necessary resources for the process. The third one will enable the process notation to support the automation of measurement and process improvement heuristics support as well as the software simulation. At the same time, various consortium’s processes and methods will be model and will have the automation support from the implementation of the notation, Environment Based Process Management.

8 References