SOFTWARE DEVELOPMENT METHOD TAILORING AT MOTOROLA

PERFECTION IS ACHIEVED NOT WHEN THERE IS NOTHING MORE TO BE ADDED BUT WHEN THERE IS NOTHING LEFT TO TAKE AWAY.
—ANTOINE DE SAINT-EXUPERY

By
Brian Fitzgerald,
Nancy L. Russo,
and Tom O'Kane

Given the pervasive nature of software in modern society, the development of software is a critical issue, but one that remains problematic. The term "software crisis" was first coined more than 30 years ago, and may be simply summarized as software taking too long to develop, costing too much, and not working very well when eventually delivered. A disciplined approach to software development through the use of software development methods could help address these problems. Hundreds of such commercial or brand-named software development methods exist, but these are not widely used in practice, and are certainly not used in their entirety [5]. Even when methods are used, the problem of means-end inversion often arises; that is, rather than focusing on the *end* (the development of software), developers become preoccupied with the *means* (the software development method) and lapse into blind adherence to low-level method steps rather than focusing on higher-level principles that make sense in the development context [5].

ILLUSTRATION BY KARINE DAISAY

In recognition of this, it is now widely accepted that methods should be tailored to the actual needs of the development context. However, there is very little by way of practical guidance to inform developers as to what steps of the method to modify or omit. Indeed, there is often a wide disparity between the official development process and the actual behavior of developers in practice [5]. To investigate this further, we conducted a detailed examination of the tailoring of the software development method at the Motorola software development facility in Cork, Ireland. From this case study of a high profile, successful software development organization, we draw lessons on tailoring that should be applicable to organizations worldwide.

Little research has been conducted to date on method tailoring specifically. However, two closely related areas are contingency factors research and method engineering.

The contingency factors approach to software development methods (for example, [1, 3]) suggests that specific features of the development context should be used to select an appropriate method from a portfolio of methods. However, one of the fundamental problems with the contingency approach in practice is that an organization is expected to have a range of methods available to developers who are expected to be familiar with each and merely choose the most appropriate one depending on the contingencies of the situation. Close familiarity with even one method is not all that common in practice among developers; thus, achieving competence with several methods is not a realistic expectation. Some of the later contingency factors research [1] recognized this fundamental flaw, and suggested a more pragmatic view, arguing that contingency be built in as a feature of the method itself. That is, rather than suggesting a repertoire of methods, the framework of the method is expected to encompass all situations.

Method engineering research (for example, [8, 10]) suggests a meta-method process whereby methods are precisely constructed from existing discrete predefined and pretested method fragments. Among the challenges posed by method engineering is that a repository that can store method fragments is required. The history of CASE suggests that elec-

The Software Development Environment at Motorola

Motorola is a major software systems provider in the mobile telecommunications sector. These systems are very large and expensive switching and communications infrastructure systems; Motorola has over 400 engineers working on software development in the Cork plant. Clients are typically large telecommunications providers who purchase Motorola systems to support their mobile phone networks. The nature of the environment in which these systems operate is one in which users take the underlying system completely for granted and expect total reliability—the so-called "dial-tone" test. Given the fact there are several significant and reputable competitors in the marketplace, the reliability of Motorola systems is critical. Also, the telecommunications technology area is constantly evolving, with new products and services continually offered. Systems are routinely adapted to incorporate interfaces to these new developments.

The systems themselves are developed using common languages such as C and C++. Technical personnel tend to have a background in engineering or computer science. Large teams of developers work on each development project and the development environment is very formalized. There is clear differentiation between the phases of design, implementation, and testing. In the case of the latter, special test laboratory facilities are available for rigorously testing each system function. The development process is also very formalized. The organization has explicitly documented their fundamental software process, the Cork Organisational Standard Software Process (OSSP). This is tailored precisely to the development process for each project and is then followed rigorously on all projects. New employees are made aware of the Cork OSSP via induction training sessions. The OSSP evolves as the company follows their program for continuous process improvement that has seen Motorola achieve Level 4 on the Capability Maturity Model (CMM). Satisfying the concepts of the CMM is important to Motorola, and a specialist group—the Process Engineering Group—exists within the Cork facility to ensure the CMM criteria is complied with. A large amount of metric data on the development process is collected and analyzed, and this information is later displayed on notice boards for the attention of developers.

Given the nature of Motorola's operating environment, and the competitive nature of the marketplace, it is vital that errors and downtime are kept to a minimum. When errors do occur in systems, a very precise process for handling the situation is mandated. All fixes undergo rigorous testing before they are released to the customer. C tronic support for such an initiative could be quite problematic.

One marked feature of both contingency and method engineering approach is they are deductive in nature in that they propose theoretical arguments as to how methods should be tailored or constructed. Very little is available in terms of practical application of these ideas in real development practice. This gap is addressed specifically in this study, which is grounded on method tailoring in practice in a large software development organization at Motorola, and is in keeping with the spirit of bridging the chasm between academic research and software development practice [7].

Figure 1. Head tk.

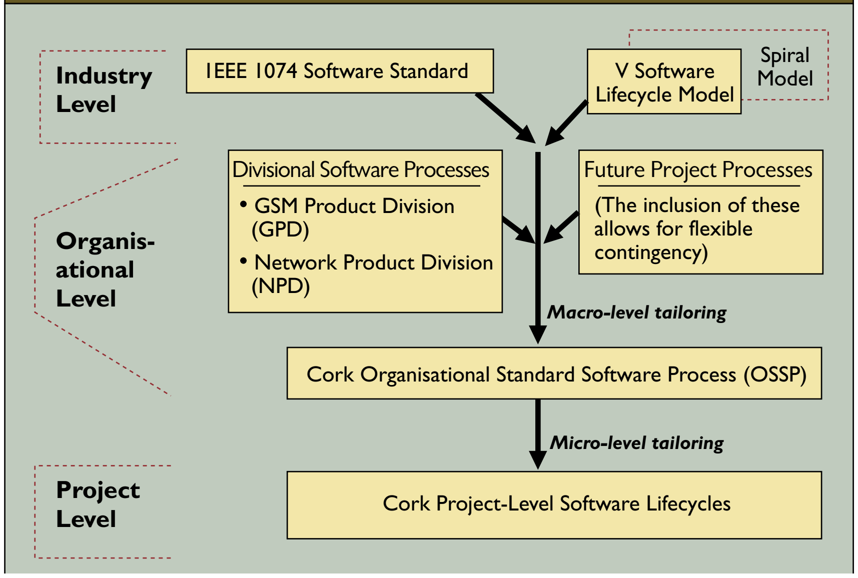
Industry Level
IEEE 1074 Software Standard
V Software Lifecycle Model
Spiral Model
Organis-ational Level
Divisional Software Processes
• GSM Product Division (GPD)
• Network Product Division (NPD)
Future Project Processes
(The inclusion of these allows for flexible contingency)
Macro-level tailoring
Cork Organisational Standard Software Process (OSSP)
Micro-level tailoring
Project Level
Cork Project-Level Software Lifecycles

Figure 1. Tailoring the software process at Motorola.

The absence of practice-based research in software development, and in method tailoring in particular, is surprising in an applied field. We chose to investigate the nature of method tailoring in practice at Motorola because it contains many lessons that can benefit both researchers and practitioners alike. A series of formal and informal interviews were conducted over a two-year period with the manager responsible for software process improvement at Motorola. More formal interviews were taped and transcribed. Informal interviews were used to clarify and refine issues as they emerged. The findings have been strengthened through the direct validation of those responsible for the process being studied.

The Method Tailoring Process at Motorola

The software development process at Motorola involves a number of discrete components (see Figure 1). These components comprise three different levels: a broad Industry level; a more specific Organizational level; and the individual Project level.

We have termed the top level of Figure 1 as the "Industry level." This reflects the fact the components are available more or less universally to any organization developing software, in that they are part of the public domain. Here, the two basic elements on which Motorola, Cork grounds its development method are the IEEE 1074 software standard [9] and the V software lifecycle model (V-SLCM, [12]). The IEEE 1074 standard is a very detailed one that prescribes a set of activities deemed mandatory for the development and maintenance of software. It comprises six high-level stages, 17 process steps, and 65 activities within these process steps. Motorola perceives a number of significant benefits in adopting the IEEE 1074 standard. Firstly, it represents an internationally recognized standard for development, one that is evolving, but in a controlled and rigorous manner. Also, the standard is complementary to the Capability Maturity Model (CMM), which is very important in Motorola as a means of assessing the maturity of their development process, and also as a mechanism to introduce improvements to that process.

While the IEEE 1074 standard is comprehensive, it merely prescribes the *processes* for the life cycle. The *products* of the life cycle in terms of specific documents and deliverables must subsequently be mapped to the method. Thus, tailoring is very much an inherent requirement of the IEEE 1074 standard.

A software life-cycle model is a time-ordered sequence of activities for development. A number of such models exist, including the traditional Waterfall, the V-model [12], and the Spiral [2]. The V-model has been chosen by Motorola to complement the IEEE 1074 standard. However, by constructing their development method from discrete components, Motorola could introduce an alternate life-cycle model if they wished; indeed, the Spiral model is also used by Motorola.

At the Organizational level, a number of software processes exist that are specific to the various parent divisions within Motorola and naturally they influence the Cork process. These divisions include the U.S.-based GSM Product Division (GPD) and the Network Products Division (NPD). Each of these divisions has configured their software process to suit their particular development environment. For example, subcontractor management is relevant to the GPD division but not to the other. Also, Motorola found some of their common software processes were not covered in sufficient depth by the IEEE 1074 standard—system testing and software maintenance issues being two examples. These needed to be factored into the organizational development process.

Finally, at the Organizational level, it is recognized that some development projects in the future might require processes that are not accommodated by the current method; hence the inclusion of the Future Project Processes component. One possible example might be a customer would seek some intermediate delivery of a product after design and prior to system testing. This would require a change to the existing processes. Thus, the existence of the Future Project Processes component ensures flexibility to cater for the contingencies that may arise in future development scenarios.

Based on these considerations, the overall Cork Organisational Standard Software Process (OSSP) is constructed. As can be seen from the earlier discussion, the development process is already characterized by a good deal of tailoring. However, this tailoring is at a macro level, and the specifics of the individual projects have not yet been factored in. At the Organizational level, the main emphasis is on creating a trusted, rigorous, and reliable software process that satisfies the sequencing aspects of a software life-cycle model (in this case, the V model). This results in a prescription for a consistent method of performing software development activities, including the sequencing of activities and the interfaces between them. The OSSP covers the development life cycle from initiation until the roll-out and close of the project. Activities that span the life cycle, such as project management, quality assurance, customer support, operational support, and training are also included.

In addition, the CMM key process areas (KPAs)[1] are explicitly factored into the method at this level. The desire to adhere to CMM criteria, which requires a standard measurable software process, motivated the development of the OSSP. The development process is measured and monitored in an extremely public manner. Throughout the Cork offices of Motorola are charts and graphs that indicate progress on various measures. All of these initiatives enforce the software method culture.

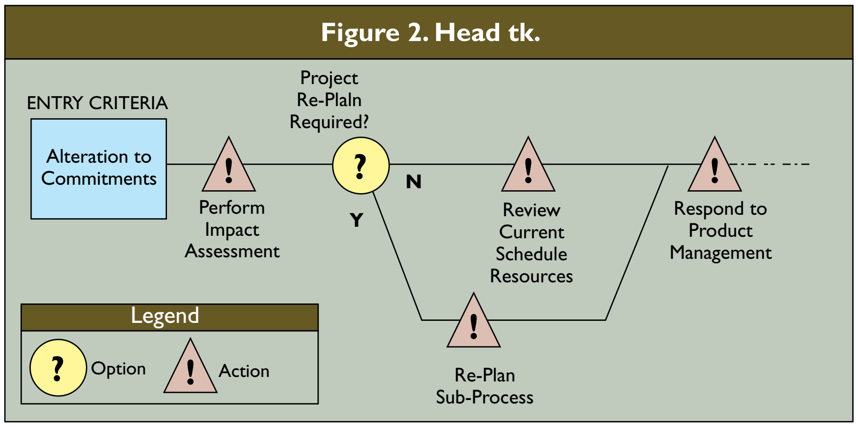
Figure 2. Head tk.
ENTRY CRITERIA
Alteration to Commitments
Perform Impact Assessment
Project Re-Plaln Required?
?
N
Y
Review Current Schedule Resources
Respond to Product Management
Re-Plan Sub-Process
Legend
Option
Action

Figure 2. Project level tailoring example.

Following the construction of the OSSP, a phase of micro-level tailoring of the method takes place at the Project level. This is where the project-specific characteristics are factored in. In essence, certain elements of the OSSP are chosen depending on the operational needs of the project. Since the OSSP elements cover all aspects of the software process, including those project-specific practices (for example, subcontractor management or project planning), and those non-specific practice (for example, training or process improvement), the project-specific elements of the OSSP must be selected to address the operational needs of the project. The Project level software lifecycle includes software standards, procedures, tools, methods, and templates. The project manager is responsible for this level of tailoring.

Specific characteristics or features of the actual project under development are then considered and further refinements to the project life cycle are duly made. Some of these tailoring decisions will be made at the start of the project and recorded in the project plan. For example, if a particular software feature is judged to be particularly complex, it may be decided to produce a High-Level Design and a Low-Level Design specification, as opposed to a simpler Detailed Design specification. Other tailoring decisions will be made dynamically in the course of the project. For example, if commitments change significantly after a project has started, then an impact assessment of the changes may cause the *Re-Plan Sub-Process* to be

[1]A full discussion of the CMM is beyond the scope of this article. However, the CMM specifies 18 KPAs central to a mature software process (see [11] for further details).

THE ABSENCE OF PRACTICE-BASED RESEARCH IN SOFTWARE DEVELOPMENT, AND IN METHOD TAILORING IN PARTICULAR, IS SURPRISING IN AN APPLIED FIELD. WE CHOSE TO INVESTIGATE THE NATURE OF METHOD TAILORING IN PRACTICE AT MOTOROLA, AS IT CONTAINS MANY LESSONS THAT CAN BE OF BENEFIT TO BOTH RESEARCHERS AND PRACTITIONERS ALIKE.

invoked again, or it may be decided to absorb the impact in the current schedule. This is further illustrated in the process map excerpt in Figure 2.[2]

The tailoring decision *Project Re-Plan Required* must be taken after the *Perform Impact Assessment* activity has been completed. The impact assessment will estimate the effect the commitment alteration is going to have on the project. This is based on variables such as code complexity, the functional area impacted, the number of interfaces affected, and staff expertise with the new requirement. The decision to re-plan will result in the Re-Plan Sub-Process being invoked. In the Re-Plan Sub-Process, artifacts such as work breakdown structures, logical priorities, and estimates will be generated, and a new schedule created. Based on the new schedule, further tailoring decisions may need to be made before commitment to the new schedule, or an alternative, is agreed. This response is then communicated to product management for ratification. Of course if the impact can be absorbed into the current schedule without re-planning then the current schedule resources will be reviewed and adjusted accordingly to accommodate the alteration.

Tailoring at this level also applies to areas that are non-project specific. For example, a change to the test process may or may not require piloting based on an impact assessment of the process change. Another example might be to grant a developer a waiver from a particular training course if the developer satisfies certain criteria.

Implications and Conclusion

We see in this case an organization that recognized both the advantages to be gained from using a standardized software development method and the need to provide a method that is tailored to fit the specific requirements of each development project. The OSSP is reasonably stable although it is expected to evolve over time, and, indeed, the capability to evolve is built into the model. It represents the general process that each project is expected to follow, encompassing the operational definition of the fundamental process elements and their interrelationships.

The tailoring strategy described here also overcomes the fundamental problem inherent in both the method engineering and contingency approaches, namely, that organizations in practice clearly cannot afford to wait while a lengthy tailoring process takes place. In the Motorola case, much of the broad macro-level tailoring is done in advance—down to the OSSP at the Organizational level. Then, at the outset of each individual project, only the precise project tailoring remains to be accomplished.

Also, this mode of tailoring does not require that individual developers possess a repertoire of software development methods from which the appropriate one is chosen—one of the main criticisms of the contingency approach. Rather, the first macro level of tailoring provides a method that is broad-ranging enough to cater for the range of development projects to be faced. This facilitates a speedy transition to the further fine-tuning necessary at the Project level. Nevertheless, the Future Project Processes component (see Figure 1) allows for the incorporation of features that may be deemed relevant. The subsequent micro-level tailoring allows for a precise fit to the unique needs of each specific project. This dual level of tailoring allows the valuable CMM elements to be incorporated, but without sacrificing any local strengths of the development process.

The modular division of the software development method into discrete prime components also has sig-

[2]The notation used in Figure 2 is based on a process-mapping and tailoring notation called PROMPT developed by one of the authors. PROMPT [6] is in widespread use within the Motorola organizations worldwide as well as several other software organizations.

nificant advantages in that individual components can be upgraded or replaced as new ideas and concepts emerge that may be worthy of investigation. The Open Source Software development model is an example of one such concept [4]. Also, an alternative life-cycle model such as the Spiral can very easily be incorporated into the standard method. Yet, the introduction of these new concepts can be implemented in a controlled and rigorous fashion. This allows the method to adapt over time to fit new project requirements. Thus the method provides both the advantages of standardization and the flexibility to cater for changes in the development environment.

In the course of this investigation several points were noted that might prove valuable to other organizations struggling to come to terms with tailoring issues:

The first of these is at the macro level. Here, the development of a project-independent standard software process, which is also independent of any particular software life-cycle model, will prove invaluable. Organizations may wish to use improvement models such as the CMM to guide this development and identify the component parts. Creating a standard software process will allow early identification and creation of the process artifacts that will be required to suit the broad macro-level tailoring requirements of each software project. If those process artifacts do not exist, then at least the organization has anticipated the need to develop them well in advance of their actual use. For example, a current project may not require subcontractor expertise, but why wait until a project has begun to discover the organization has no process, procedures, tools, templates, and so on for handling sub-contracted software?

There are essentially two types of tailoring at the Project level: tailoring planned up front and recorded as part of the project plan and dynamic tailoring invoked as the project geometry changes, as is typically the case in software development, for example, when additional requirements are discovered. For both types, the capture and definition of the tailoring criteria have proven essential for risk management and overall project success. The sources for these criteria are usually rich and varied and can include previous project records, project post-mortem minutes, brainstorming sessions, customer satisfaction surveys/feedback, participant observation, and decision analysis. Such information can be captured in a database or, in the case of Motorola, Cork, on Web-based process maps. This information can be treated as living documentation, as it is constantly being reviewed and augmented as the organizational capability develops. Establishing clear Project level tailoring criteria will provide a baseline to ensure a consistency of approach to both project planning and execution-time contingency planning.

While this type of predefined tailoring may not be possible in all organizations, this case does illustrate that tailoring is necessary and feasible, even in rigidly controlled development environments. Once an organization reaches the point where it can identify the various characteristics or contingencies that occur in its development projects, then it is possible to build flexibility into the method, along with the rules to allow developers to identify appropriate choices. Certainly, the evidence suggests that Motorola has gone some length to effectively address the problems of the software crisis, and continues to pursue the path of continuous improvement. This has resulted in the organization being independently certified for various industry standards including CMM and ISO9001. C

References

1. Avison, D. and Wood-Harper, A. Information systems development research: An exploration of ideas in practice. *The Computer J. 34*, 2 (1991), 98–112.
2. Boehm, B. A spiral model of software development and maintenance. *IEEE Computer 21,* 5 (1998), 61–72.
3. Davis, G. Strategies for information requirements determination. *IBM Systems J. 21,* 1 (1982), 4–30.
4. Feller, J. and Fitzgerald, B. *Understanding Open Source Software Development*. Addison-Wesley, UK, 2002.
5. Fitzgerald, B. The use of systems development methodologies in practice: A field study. *The Information Systems J. 7*, 3 (1997), 201–212.
6. Fitzgerald, B. and O'Kane, T. A longitudinal study of software process improvement. *IEEE Software*, May/June (1999), 37–45.
7. Glass, R. Is criticism of computing academe inevitably divisive? *Commun. ACM 42*, 6 (June 1999), 11–13.
8. Harmesen, F., Brinkkemper, S. and Oei, H. Situational method engineering for IS project approaches. A. Verrijn-Stuart and T. Olle, eds. *Methods and Associated Tools for the IS Life Cycle. Elsevier Science*. North-Holland, 1994, 169–194.
9. IEEE Standard for Developing Software Life Cycle Processes (1991). IEEE Computer Society, New York, NY.
10. Kumar, K. and Welke, R. Methodology engineering: A proposal for situation-specific methodology construction. W. Cotterman and J. Senn, eds. *Challenges and Strategies for Research in Systems Development.* Wiley & Sons, Chichester, UK, 1992.
11. Paulk, M., Curtis, B., Chrissis, M. and Weber, C. Capability Maturity Model for software 1.1. *IEEE Software 10*, 4 (1993) 18–27.
12. Sommerville, I. *Software Engineering.* Addison-Wesley, UK, 1992.

Brian Fitzgerald (bf@ul.ie) is Frederick A. Krehbiel II Professor of Innovation in Global Business and Technology at the University of Limerick, Ireland.
Nancy L. Russo (nrusso@niu.edu) is an associate professor and chair of the department of operations Management and Information Systems at Northern Illinois University, DeKalb, IL
Tom O'Kane (tokane01@motorola.com) is Distinguished Member, Technical Staff at Motorola, Cork, Ireland.