Introducing Agility: A Case Study of Situational Method Engineering Using the OPEN Process Framework

M.K. Serour  
University of Technology, Sydney  
mserour@it.uts.edu.au

B. Henderson-Sellers  
University of Technology, Sydney  
brian@it.uts.edu.au

Abstract

Contemporary software development organizations have a need for flexible processes and methodologies. They eschew formal, heavyweight methods for the use of more agile methodologies. Creating such an agile methodology is shown to be facilitated by using the theory of situational method engineering. This is then exemplified in a case study in a well-established, multinational legal publishing company in Australia.

Keywords: object technology, agile methodologies, method engineering, case study

1. Introduction

The role of software methods/processes in supporting software projects is being realized as a valuable tool for controlling the ever-increasing complexity and size of today’s commercial software systems. With the growing adoption of new business approaches such as e-business and globalization and also the increasing diffusion and utilization of new technologies such as agents, Internet and web services, both software managers and developers are in desperate need of effective and appropriate methods or processes to help them in their activities of producing trustworthy software products to support these new business approaches and best utilize new technologies. Consequently, software projects are expected to be very different in domain, scope, size and type. Hence, selecting one single method or process for all these different software projects, even at the organizational level, seems impractical and infeasible. Once an organization decides on what software products they need, the main challenge remains in selecting the most suitable method to best manage their software project.

In this paper, we first investigate the choices available in existing methods followed by our proposal of using situational method engineering with the OPEN Process Framework (OPF) in order to construct an organizationally specific agile method/process, tailored to be optimal for each individual project. This method engineering proposal is based on theoretical research as well as several industrial experiments. We begin (Section 2) with an evaluation of the options for selection of an industrial-strength software development methodology, concluding that a good solution uses the ideas of situational method engineering (SME) (Section 3). As an exemplar of SME, OPF [1,2], is then described in Section 4 and its use expounded in Section 5 for an action research case study of an international legal publisher in which we focus on their use of SME, complementing the social and organizational results previously published [3,4].

2. The Process Choices We Have

The terms “methodology” and “process” are often interchanged and have no unique definition in software engineering. We define process here as a set of practices (often sequential) that are functionally coherent and reusable for the organization, implementation and management of software [5]. A methodology (or method) on the other hand has two components: a process element (as above) and a work product-focused element. To be accepted as a formal methodology, a software process must be documented, repeatable, reachable (i.e. usable), based on proven techniques, have been validated and be appropriate to the problem to be solved [6].

In this paper, we will be focussing on the process constituent parts of a methodology and hence, despite the above differentiation between process and methodology, we will in fact use the terms process and methodology as synonyms except where the difference (viz. the inclusion or exclusion of a product model) is important.

Creating an actual process requires instantiating a process metamodel i.e. creating actual process components from their standardized
conceptualizations. However, in doing so, there are several options relating to how decisions are taken during that instantiation process.

We can make almost no decisions and simply document the instances of each metaclass together with rules for how they relate to each other. This creates a set of process components and is the framework approach exemplified by OPF [1]. Alternatively, we might prescribe a particular lifecycle model (contract-driven, waterfall, spiral) and one or more specific mandatory techniques (such as in the use-case driven approach of RUP [7] or in the Shlaer/Mellor [8] translational approach).

In the first case, as when using OPF, decisions need to be made by the user rather than the methodology designer. This has the distinct advantage of supporting a high degree of tailorability by the end user so that an organization can choose exactly the components of the OPF that suit them and hence construct a methodology optimal for their local conditions (so-called “OPEN variants”). However, this is balanced by the disadvantage that this is extra work (Figure 1).

![Diagram showing tailorability axes]

2.1. Pitfalls of Selecting a Predefined Method

Based on industry projects, it is generally accepted that the creation of one single method to manage and support every software project is not achievable. Any predefined methodology will be useful but will not entirely match the organization’s methodology requirements. Indeed, software projects change very frequently to a degree that the deployed software method must be responsive enough to respond to any changes promptly and effectively. At the organizational level, moving from one project to another implies the re-application of selecting an optimal method for the next project. This vital activity of method selection is most likely to be repeated for every subsequent project. Thus, the actual methodology in use is not stable – it is different for each project. Thus, the costs of learning and possible cultural shift are potentially very high.

In addition, the idea that business culture should be adapted to fit a specific methodology is not good business sense, despite its prevalence in many of the marketed methodologies to date. When using IT and its dependent parts, such as methodologies, it is critical that they fit the business and not the other way around.

2.2. The Solution: Create Your Own or Do It Yourself

The traditional concept of processes, in which a single process can suit all software projects, has proved to be false since, as noted above, software projects vary greatly depending on many factors, such as organizational constraints, business constraints, technology issues and application factors [2]. Also, the fast-changing development environment and more specific user requirements demand more flexibility from processes in order to satisfy the demands created [10].

Hence, there is the need for a more generic and universal approach, i.e. the notion of a process framework. A process framework consists of a metamodel and a repository of process components, used to create processes to suit the specific characteristics of the situated project. This is called “situational method engineering (SME)” e.g. [118]. Furthermore, construction of a specific process instance requires guidelines for (i) selecting and (ii) assembling the process components from the repository since every organization and every software project has a unique set of characteristics. Encapsulating these guidelines in a CASE/CAME (Computer Assisted Software Engineering/Computer Assisted Method Engineering) tool [19,20] provides a step towards automation of SME.

---

**Figure 1. Range of “tailoring” for four examples (OPEN, OPEN variants, RUP and Shlaer/Mellor). Note that since first publication [7], RUP has been extended and modified significantly towards the left-hand side of this axis (after [9]) ©SIGS Publications**

In the second case, the methodology designer makes all the decisions for the user and mandates every step and every item in the methodology. Thus the resulting predetermined methodology is only usable “out of the box”. There is no need for tailoring but, at the same time, there is zero flexibility, although if it exactly matches the requirements of the organization, then this is the perfect answer [9]. Finally, most of the predefined contemporary software methods fall into two main categories, heavyweight or formal and agile.
3. The Situational Method Engineering Approach

Situational method engineering involves defining a repository of method fragments together with techniques for assembling these method fragments into site-specific methodologies specifically tuned to the situation of the project at hand [11] i.e. one that meets the requirements of a particular project. Thus, selection of method fragments is individualized and “tailored” to the specific requirements of the organization and project. Many papers describing situational method engineering tend to focus on the process engineering element rather than the combination of process and product viz. the “methodology”. In other words, they, as do we, take process engineering and method engineering effectively as synonyms.

Over recent years, there has been significant research into the field of method engineering and situational method engineering [11,13,21,22]. The work in [12] and [14] is worth mentioning. These authors focus on the techniques for assembling available method fragments into situational methods/processes – an approach where a sound mathematical basis for process construction is described formally in a number of rules. The use of these rules allows the creation of a knowledge base, where knowledge can be stored and disseminated.

The cultural issues of adopting and inculcating any methodology into a software development organizational unit have not been well studied. Introducing a formal, heavyweight approach in a single adoption project [23] is unlikely to be successful [3]. Instead, the current authors have been investigating incremental adoption in an action research study [24]. The adoption process begins with a small methodology and incrementally and iteratively creates a more and more sophisticated one. The methodology and the developers’ ease with using the methodology grow concurrently. Organizational and social/cultural issues have been analyzed for the legal publishing company [3,4]; here, we focus on the technical aspects of situational method engineering with the OPEN approach to underpin this iterative growth of the company’s tailored OO methodology.

Based on this body of ME theory and our experimental study, we thus advocate the incremental adoption of an ME mindset with a shift in the way we think about the selection and deployment of a software development method. We confidently promote the adoption of a situational method engineering approach in which a customizable, personalized methodology is constructed from a rich repository of method chunks, rather than the selection and use of a predefined single method. In the following sections, we introduce the situational method engineering as an effective approach for constructing an organizational method that may be tailored or customized for individual projects in the context of the OPEN process framework.

4. OPEN: The Process Framework

Unlike other OO software processes, OPEN [1,2] is defined as a process framework defined by a metamodel. It is thus highly compatible with the ideas of method engineering and process construction as described above. From this framework, OPEN-compliant processes can be instantiated to be used in actual organization and software projects. In other words, the process engineer has to configure OPEN, creating an instance of OPEN that is suitable for use on a specific project. Instantiating OPEN is one of the more difficult and time-consuming jobs in adopting OPEN, since the process engineer has to understand the methodology, the organization, the environment and the software project itself in order to select the appropriate components in the OPEN repository to use on the project. Traditionally, this process is carried out using predefined organizational requirements and the experience and knowledge of the process engineer although tool support is likely in the near future [19].

An important part of OPEN is the repository of process components, which can be used in different software projects. This repository provides an almost complete set of components and can be extended to support changes in technology.

Figure 2. The five major metaclasses of OPEN's metamodel (after [2]) © Addison-Wesley.

The OPEN metamodel defines five main classes of process components as shown in Figure 2. From these (and their subclasses) are generated a significant number of instances, together with a number of
5.1 Company Background

Our empirical study for situational method engineering was conducted in a well-established, multi-national legal publishing company and a leading publisher of legal and tax information to the professional services industries.

In the past, the company’s focus had been on the production and distribution of printed material; about four years ago, a decision was taken to transform the organization into a provider of on-line information in its domain, requiring a transformation of the business from its traditional practices to an electronic business environment. This transformation has meant that the IT and business strategies of the organization are now one and the same; the organization as a whole is dependent on its capability of delivering effective systems.

The IT division within the study organization was established as a key element in realizing the organizational transformation operating from two sites in Sydney and Canberra. It has been developing software since early 1995 when the development team consisted of two programmers, a few business analysts and a database designer. Since then, the division has grown and now has around 60 people, most of whom are involved in some form of software development. Not only is this a massive increase but it is also compounded by the fact that there are two very different types of development: large systems development and web development. The IT division has been responsible for the development of an integrated publishing environment focussed on the management of large text databases, using Oracle and Java. The division is seen as a commercial consulting division, judged by the same criteria as the other arms of the company. The publishing sections of the company are the clients of the division; they develop budgets for technical work and commission applications from the division.

The development team of the IT division is made up of a mixture of experienced and inexperienced permanent staff and a number of contractors. All software development is done in a fairly ad hoc way, often with tight deadlines, which has caused major problems especially with contractors. These problems manifest themselves in a number of different ways: mismatches in communications when discussing different aspects of the system, time wasted due to ad hoc/informal on-the-job training and inconsistencies in the final design.

5.2 Compelling Reasons for Change

Following the senior management decision to move to globalization, the study organization was going through a major business transformation process from traditional business practices to a totally electronic business environment (e-transformation). It was clearly observed that the entire organizational changes were driven by strong compelling business reasons vital for the organization to remain competitive in the new global e-market. One of the major organizational aspects of the transformation process to e-business is the transitioning of the internal IT division to an object-oriented development environment and the adoption and diffusion of an appropriate software development method that best suits their new e-business development environment. IT division’s customers (other internal departments) required that software systems and the organization’s web sites must be able to adapt more easily to changing business needs, can meet their unique requirements and can scale to large and small installations.

5.3 Assessing the Existing Options

Initially, the organization management decided to conduct a formal assessment of the contemporary OO software development processes/methodologies to choose an appropriate method to be adopted and diffused by their IT personnel. This assessment task was commissioned to an outside third-party consulting firm in collaboration with a small senior management team. With a strong recommendation of some biased consultants and senior managers, the organization decided to adopt one of the most rigid, formal and heavyweight methods available. Soon, management realized the unsuitability of such a heavyweight methodology. Furthermore, IT personnel discovered the inappropriateness and the impracticality of the chosen method due to its inflexibility for customization for their diverse projects. Everyone felt that the
selected method had too much overhead, was resource hungry and far ‘heavier’ than necessary. As a result, IT personnel showed resistance to change in the form of doubting the benefits of the change, developing excuses not to participate in any transition activities and showing their fear and uncertainty of the final result.

5.4 Understanding the Existing and Desired Culture

In an attempt to manage the evolving IT personnel’s resistance to change, one of the IT developers (RH) who was undertaking a postgraduate course at the local university (UTS) approached the Centre for Object Technology Application and Research (COTAR) at UTS to assist them in their transitioning process to an OO environment and the adoption or construction of a new method for their software development. Such a method needed to be flexible enough to be customized to suit small and large projects. The first author of this paper (MKS) was appointed to work with the organization through their transition process for 1-2 years full time as a researcher and co-practitioner using the action research methodology. The second author (BH-S) provided the organization with a number of education and training courses on object technology and project management.

With the senior management approval, MKS with other three IT personnel formed a transition champion team to lead and promote the transition process. The champion team had an initial meeting with the senior IT manager to work out a strategy for the transition process. During that meeting, the champion team emphasized the fact that understanding both the existing and desired organizational and human work culture was a mandatory starting point [25]. Therefore, a number of face-to-face meetings, with a video conference with the secondary location, were conducted to identify and understand the existing and desired cultures. Management insisted, and encouraged everyone within the IT division along with their internal customers, to participate in that assessment whether by attending the assessment sessions or by providing their culture details and suggestions/feedbacks. In addition, the IT division was assessed in terms of their existing software development process by a third-party consulting firm using the SPICE (Software Process Improvement and Capability dEtermination) [26] model with its rapid organization assessment technique. During SPICE assessment meetings, the IT personal work culture was discussed in great detail, which helped everyone to fully understand the existing culture. Examples of the IT work culture examined included their existing processes for requirements elicitation, software development and problem resolution. Also, their existing management culture was investigated for further understanding and assessment. IT management culture includes project, configuration and risk management. Table 1 shows the SPICE assessment level achieved for a number of their process areas. Their software development maturity was assessed as being roughly equivalent to level 1 of the CMM (Capability Maturity Model) [27] scale. The organization declared an aim of achieving, at least, level 3 of the SPICE scale for their various process areas (roughly equivalent to CMM Level 3) by the end of the transition project. This allows the maturity of the existing software processes to be assessed and key practices identified that are required to increase the maturity level.

Table 1. Results of the SPICE assessment undertaken prior to commencement of the IT transition process

<table>
<thead>
<tr>
<th>Process to be assessed</th>
<th>SPICE level achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements elicitation</td>
<td>1</td>
</tr>
<tr>
<td>Software development</td>
<td>1</td>
</tr>
<tr>
<td>Configuration management</td>
<td>0</td>
</tr>
<tr>
<td>Quality assurance</td>
<td>0</td>
</tr>
<tr>
<td>Problem resolution</td>
<td>0</td>
</tr>
<tr>
<td>Project management</td>
<td>1</td>
</tr>
<tr>
<td>Risk management</td>
<td>0</td>
</tr>
<tr>
<td>Process establishment</td>
<td>0</td>
</tr>
</tbody>
</table>

This was a good starting point for everyone within the IT division to understand their existing work culture and their current process for software development. Based on the analysis of the SPICE report, both management and development teams were able to identify the strengths and weaknesses of their current work culture as well as understanding the new, desired OO work environment to be achieved through professional education and training sessions.

As well as an acknowledgement of the current state, it is also important to set goals for the transition. It is necessary to assess these requirements in order to determine and estimate the required resources to carry out the transition. Further assessment activities are focussed on assessing people in terms of their skills, knowledge and willingness. The whole team/organization will not transition in a single “big bang”, so careful planning is needed for a phased or
incremental transition. One way of setting the goal would be in terms of forecasting the final CMM or SPICE level that might be achieved following a successful transition exercise. Therefore, the champion team with the IT director put together a plan to manage the necessary culture change to achieve the following objectives:

1. Change the current work culture to a new one where OO is dominant.
2. Improve the existing software development practice by adopting and diffusing a formal OO software engineering process with an appropriate modelling language and CASE tool.
3. Provide adequate and professional education and training to master the new technology.
4. Achieve level 3 on CMM or an equivalent level on SPICE scale.
5. Plan for continuous improvement.

5.5 The Adoption of the Situational Method Engineering Approach and the OPEN Framework

Following an initial management approval, the project started with conducting a number of education and brainstorming sessions for both management and developers to explain and discuss a number of vital issues including the requirements of their desired state, the agile software development approach, the OPEN process framework and the fundamentals and advantages of adopting the method engineering approach to fulfil their requirements. The major reason for proposing the use of the OPEN process framework was clearly explained. As noted above, OPEN has an underpinning metamodel. There are elements in the metamodel to describe process elements such as Activities, Tasks and Techniques; people elements such as Producers and Roles; organizational elements, such as Enterprise, Programme and Project, and product elements in the form of a whole range of Work Products including diagrams and documents, supported by various kinds of languages (natural language, modelling language and coding language). Consequently, the organization, with everyone’s consensus, decided to accept the UTS COTAR proposal to engineer their own software development method. Furthermore, management formed a new team called the ‘method engineering team’ from a number of IT people, two representatives of their customers and the organization IT director. This team was formed in such a way as to achieve some imperative goals including giving IT people the feeling of the ownership of their method, enforcing customer involvement and representing management commitment, all considered as critical success factors. For the following eight months, both champion and method engineering teams worked in close collaboration with everyone involved including customers to construct a flexible method to satisfy the requirements of each and every project of their organization. Finally, the organizational agile method was created, along with a number of easy-to-read documents. It was well accepted and is cherished by both management and technical people. Figure 3 illustrates the software development method that was engineered for the study organization. The method engineering process started by adopting one of the predefined examples provided by the OPEN guidelines as a base start. Subsequently, the method engineering team conducted a number of brainstorming sessions with IT personnel and their customers to modify the adopted method to suit their organizational needs. This was a good start for the IT teams to learn how to customize their own method to suit different projects by plugging and unplugging process components. The engineered method went through a number of iterations to reflect all people’s concerns and needs until the first version was released towards the end of the year.
5.6 The Role of Method Engineering in Culture Change

Adopting the situational method engineering approach enabled IT people within the study organization to create their own method in an incremental and iterative manner following the concept of the “small wins” technique, a technique observed to be successful [4] and proven to be very effective in changing people’s culture and managing their natural resistance. In addition, adopting SME helped IT professionals to get familiar and comfortable with their new OO method. This again resulted in enhancing their capabilities and also in reducing their total resistance to change. The process of method engineering has changed everyone’s perception of the main role of a software development method. It was clearly noticed that everyone was convinced that the new method is neither to measure people’s productivity nor to track their performance. Rather, the deployment of an appropriate method with associated tools, such as a CASE tool, is a good communication tool between development teams, management and end users. It was also clear to everyone that the main role of such a method is to be used as a road map to provide them with the best direction, rather than a recipe book that they have to follow step by step. This perception and understanding have given everyone the feeling of self-confidence and ability to follow the new method with less fear. It also helped them to feel more comfortable and familiar with the new work environment. Another observed advantage of SME was the new people’s feeling that the method is an organizational property and doesn’t belong to any individual or a group of individuals within the organization; rather, it is owned by everyone. Ownership feeling considerably helped the management in eliminating people’s resistance to change. They have shown their willingness and readiness to participate and support their transition to the new software development environment with no fear and less uncertainties.

Figure 3. The engineered method for the study organization
6. Conclusion

In the context of contemporary software development, we have presented the theory of situational method engineering (SME) and one particular approach to this using OPEN [1]. Based on an extensive, action research case study with an international publisher moving to object-oriented processes in a web environment, we have shown how a organization- and project-specific methodology can be created from the OPEN framework and how, by using the approach of incremental adoption of this new methodology, they have been successful in creating a culture within their organization and their development teams commensurate with modern-day information technology.

7. References