Using the OPEN Process Framework to Produce a Situation-Specific Requirements Engineering Method

D. Zowghi[1], D.G. Firesmith[2] and B. Henderson-Sellers[1]

[1]University of Technology, Sydney, PO Box 123, Broadway, NSW 2007 Australia ({didar,brian}@it.uts.edu.au)

[2]SEI, Carnegie Mellon University, Pittsburgh, PA, USA (dgf@sei.cmu.edu)

Abstract

Since it is not possible to identify or to create a single method that is appropriate for all situations, the need for a focussed requirements engineering method (REM) necessitates the search for a mechanism that will support the flexible creation of a number of tailored REMs from a single base. Using a repository of reusable method components, it is possible to use the techniques espoused by the method engineering community to construct an appropriate REM that is well-suited to the particular system or application development endeavour under consideration. One particular example is used to illustrate this approach – that of the OPEN Process Framework (or OPF).

Keywords:

Method engineering; requirements engineering; process construction; OPEN

Submission category: Regular paper

1. Introduction

"A process model is an abstract definition of an actual or proposed process" [8]. A process model, also called a method here, provides the "textbook description" of all the elements that should be enacted on a real project. That enacted process model is called the process and is the focus of, for instance, software process improvement (SPI).

Here we focus on a process model/method for requirements engineering. Expanding on the above definition, we can say that a Requirements Engineering Method (REM) is a structured and coherent set of tasks, procedures, work products, policies, organisational structures and technologies needed to identify, analyse, specify, validate and manage a high quality set of requirements. In practice, Requirements Engineering (RE) is an iterative process, whereby requirements emerge and evolve in an iterative incremental rather than a sequential manner [11]. A complete REM description should include statements about what tasks are carried out, the structuring or scheduling of these tasks, who is responsible for each task, the inputs and outputs to/from the tasks and the tools used to support the method when it is enacted as a process for a particular project [35].

In the RE literature, different definitions have been given for this method and its tasks. In some cases, an REM is defined at a very fine level of detail and the steps in the method must be carried out (enacted) exactly as described. However, this form of process model description usually applies to very simple processes; for more complex processes, the description is usually less detailed and it is up to the person or project team who are executing or "enacting" the process to carry it out in their own environment. Furthermore, an REM includes tasks involving individuals as well as groups and, as such, is inherently susceptible to problems arising from human-related issues. It is thus difficult to write down a generic sequential plan of tasks that adequately describes the endeavour-specific REM.

An REM typically begins with *elicitation* followed by *modelling* and *analysis*. The results are then formalised as different kinds of requirements, which are documented into one or more requirements *specifications*. This is followed by *verification* against characteristics of good requirements (e.g., completeness, correctness, lack of ambiguity, and feasibility) as well as by *validation* against stakeholder needs and desires. *Management* of requirements is considered as a continuous activity throughout the development lifecycle, within which the integrity and consistency of the requirements model are maintained. An REM thus exploits a number of fundamental elements:

- RE tasks and techniques – Various techniques and procedures exist within RE research and practice for each of the tasks in the RE method [9].

- RE tools – In order to perform RE tasks effectively, a number of commercially available tools have been developed (e.g. DOORS, RequisitePro, CaliberRM, and CORE).

- Organisation and people – RE is carried out by teams of people playing various roles that have to be coordinated and managed within an effective organisational structure [22,25].

- Programmatic factors – Different tasks of an RE method must be shaped in such a way as to properly take into account the size of software, its complexity and the context where software is supposed to be sold and used [1].

Viewing the development of requirements work products (e.g., system and software requirements specifications) from a process viewpoint helps to identify the different dimensions of RE and the problems that need to be addressed in order to establish effective RE practices. Indeed, addressing the issues and challenges of REM is not a matter of introducing a new tool and environment or merely selecting or devising a RE process model. Instead, attention should be paid to the complex interplay between a number of organisational, cultural, technological and economical factors impacting the RE process.

Very few organisations have an explicitly defined and standardized RE methods and mostly define the product of the process, typically a software requirements specification SRS [23]. Clearly, organisations will benefit from understanding their RE processes and defining an REM that is appropriate to their organisational needs and specific software projects in which they are engaged. Indeed, it is generally acknowledged [7,19] that, at least at the full lifecycle granularity, it is not possible to identify or construct a single method that results in a process that is appropriate for all situations. Consequently, the approach of method engineering [3,4,17,24,28,29], as we shall demonstrate, offers valuable insights and tools by which to create a tailored requirements engineering method that is highly suitable for the specific, identified endeavour (e.g., project or programme of related projects). Here, we encapsulate the ideas of method engineering (ME) within an object-oriented framework, the OPF (OPEN Process Framework) [13]. The OPF is a standardized approach, originally devised for full system lifecycle methodology creation [16], but here evaluated in terms of its capability of offering adequate support for the creation of a *requirements engineering method*. This set of guidelines presented here offers ME ideas on how to use/tailor a method framework to produce a more generic RE method. These guidelines could equally be applied to any method framework, any organisation and any project.

2. The OPEN Process Framework (OPF)

2.1 Introduction

The OPEN Process Framework (OPF) consists of three major parts (Figure 1):

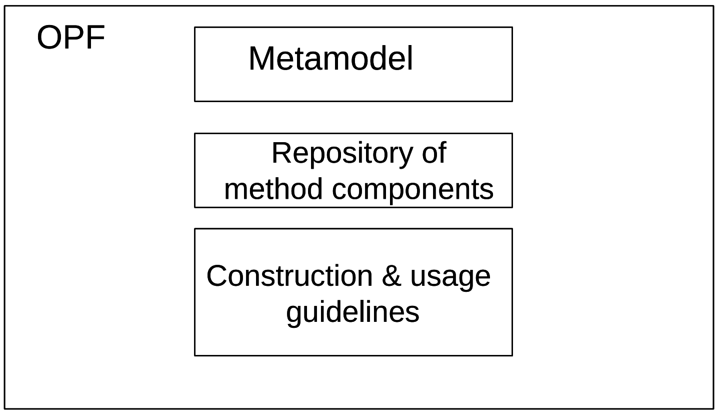
OPF
Metamodel
Repository of method components
Construction & usage guidelines

Figure 1. The main elements of the OPF

- A **metamodel** defining the fundamental kinds of reusable method components and how they are related to each other.
- A **repository of reusable method components** (actual descriptions of each kind of reusable method component)
- **Construction and usage guidelines** on how to reuse the method components in the repository to produce situation-specific processes.

These are described in the next three sub-sections.

2.2 The OPF Metamodel

The OPF metamodel provides a standard terminology and semantics for the elements in the repository of free open source reusable method components. Based on the elements in the metamodel (Figure 2), the method components fall into a small number of major groupings:

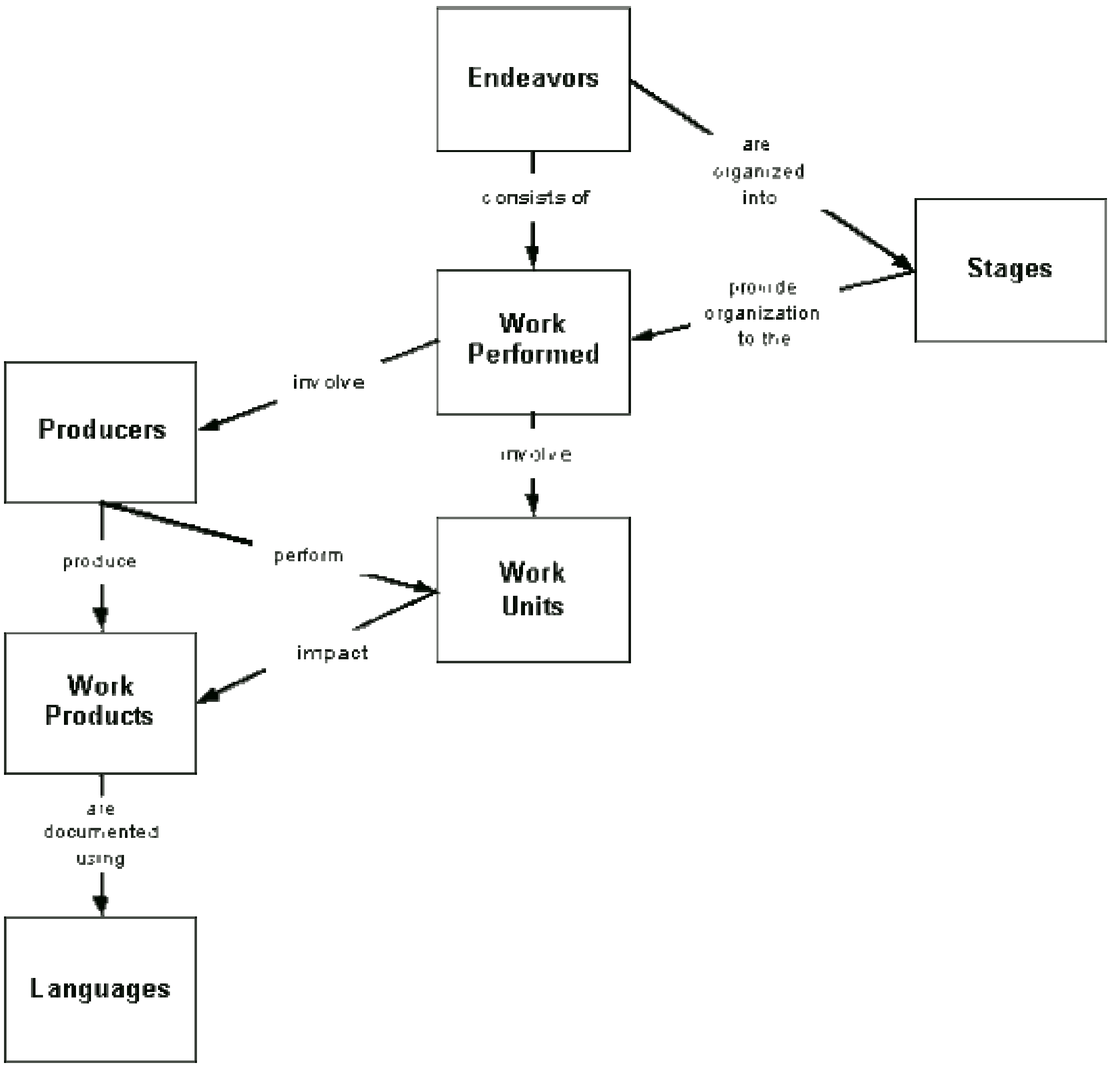

Figure 2. The major meta-elements of the OPF. All method components are generated as instances of one of these meta-elements and documented in the repository.

- **Work Products** are method components that model anything of value (e.g., documents, diagrams, applications, classes) produced by the collaboration of one or more producers during the performance of one or more work units.
- **Work Units** are method components that model functionally cohesive operations that are performed by producers during the delivery process. OPF recognizes the following three kinds of work units:
  - **Activities,** which are the highest-level of work units consisting of cohesive collections of one or more tasks that are performed by one or more collaborating producers when either producing a set of one or more related work products or when providing one or more related services. For example, requirements engineering is an OPF activity.

    - **Tasks**, which are mid-level work units that model a functionally cohesive operation that is performed by one or more producers. For example, requirements elicitation, requirements analysis, requirements specification, requirements validation, and requirements management are OPF's primary requirements engineering tasks.
    - **Techniques** are low-level work units that model the way that one or more tasks are performed. For example, use case modelling would be a technique for engineering functional requirements and hazard analysis would be a technique for engineering safety requirements.
- **Producers** are method components that model anything that produces, either directly or indirectly, versions of one or more work products. OPF recognizes the following kinds of producers: organizations, teams, roles, tools and persons. For example, requirements team and requirements engineer are two OPF producers.
- **Languages**, which are method components that model the languages used to document work products. For example, UML could be used to document use case models and Object-Z could be used to formally specify requirements.
- **Endeavours**, which are method components that model large-scale ventures undertaken by collaborating producers during multiple stages to develop and/or maintain one or more related applications. The OPF metamodel defines the following subclasses of endeavours in the OPF repository: projects, programmes of related projects and enterprises. In the REM, this gives a contextual setting only.
- **Stages**, which model formally identified time periods or points in time that provide organization to the work units of the delivery process. Typical kinds of stage are cycles, phases, builds and milestones. For example, you could define a milestone marking when the requirements for a development iteration will be complete, under configuration control and frozen.
- **Work Performances**, which are method components that model work units as performed by producers.

2.3 OPF Repository of Method Components

The OPF contains a repository of free, open source reusable method components. These are used by following the tenets of Method Engineering [3,24]. In this approach, a "personalized" method is created for a specific organization, a specific division or a specific project by bottom-up construction from a number of these method components [4], here identified by the OPF metamodel and using the construction and usage guidelines (Section 2.4) in order to aid the actual construction of the REM.

In Section 2.2, only the types of the method components are listed. For use on an actual project, each of these types and subtypes is used to generate (by instantiation) a wide range of *actual* work products, techniques, activities, roles etc. Those relevant to the construction of an REM are summarized below.

The only relevant OPF Activity is, naturally, that of Requirements Engineering, although there are subclasses of RE such as RE for developing a system, RE for developing a software application and RE for developing the reusable requirements for a specific application domain. Requirements engineering typically involves teams and roles performing requirements tasks in an iterative, incremental, parallel, and time-boxed manner. From the OPF repository, method components can be identified for each of these[1]. Useful RE Tasks include

- **Stakeholder profiling** is the task during which the representatives of all major stakeholders of customer organization's current business enterprise of the are studied, modeled, and analyzed.

- **Customer analysis** is the task during which the current business enterprise of the customer organization is studied, modelled and analyzed.

- **Competitor analysis** is the task during which competing businesses of the current business enterprise of the customer organization are identified, profiled, studied, and analyzed.

- **Market analysis** is the task during which the current or planned marketplaces in which the business enterprise of the customer organization are identified, studied, modelled, and analyzed.

- **User analysis** is the task during which the current and future intended user organizations of the application(s) of the customer organization's business enterprise are identified, studied, modelled, and analyzed.

- **Business visioning** is the task during which the customer organization's vision of their [re]engineered business enterprise is produced and documented.

- **Application visioning** is the task during which the customer organization's vision of a new or updated application is produced and documented.

- **Requirements elicitation** is the task during which raw new potential requirements for the business enterprise are identified and captured.

- **Requirements analysis** is the task during which elicited and reused requirements for the business enterprise are studied, modelled, refined, prioritized, scheduled, and traced.

- **Requirements specification** is the task during which requirements, requirement diagrams, and requirements models for the business enterprise are documented in requirements specifications and related documents.

[1] For further details see [12] and http://www.donald-firesmith.com/HowToUse/REPF.html].

- **Requirements reuse** is the task during which reusable requirements and requirements-related analyses are identified, evaluated for relevancy, and where appropriate reused (possibly with modification).

- **Requirements management** is the task during which the storage, access, approval, publication, and tracing of requirements work products are managed.

- **Technology analysis** is the task during which the potential technologies for future applications are identified, analyzed, and documented

- **Requirements prototyping** is the task during which one or more prototypes are produced in order to identify and iterate requirements

To facilitate these Tasks, there are many possible documented Techniques. Some of the most useful are:

- Abstraction – to ensure the correct level of detail in the requirements

- Brainstorming – to rapidly identify stakeholders and their requirements

- Documentation standards – to create requirements specifications of high quality

- Documentation templates – to generate skeleton requirements specifications

- Gap analysis – to analyse differences between capabilities of current application and the needs of future versions

- Inspection checklists – to evaluate the quality of the RE work products

- Interviews – for eliciting requirements from stakeholders

- Joint application development (JAD) – to elicit requirements by the involvement of stakeholders, especially customer representatives

- Prototyping – to create user interfaces initially to get an early customer reaction

- Questionnaires – to elicit stakeholder needs and requirements

- Reference requirements – to be used as a standard and baseline across multiple endeavours

- Requirements patterns – to find standardized solutions to commonly occurring RE problems

- Storyboarding – to elicit and understand user requirements

Typical Work Products for the REM include:

- Software Requirements Specification (SRS) – documents a cohesive set of requirements, possibly including their associated models, diagrams, and ancillary information

- Use case diagram – depicts functionality expressed in terms of use cases

- Class diagram – depicts definitions of objects (classes)/concepts and their inter-relationships

- State transition diagram – depicts for a single entity or class the various states it can be in and how changes of state can be triggered

A language is set of terms combined with associated syntax and semantic rules that are used to produce one or more work products. Whereas programmers are primarily concerned with various kinds of implementation languages, requirements engineers and their technical writers use the following kinds of languages to implement requirements work products:

- Natural languages - such as English, although often ambiguous, are most often used to specify textual requirements,.

- Modelling languages - such as the Unified Modeling Language (UML) and the Object Modelling Language (OML), can be used to produce requirements diagrams and associated graphical models.

- Specification languages - such as Z and Object-Z, are sometimes used to specify requirements more formally and so that they can be verified using tools such as theorem provers or model checkers.

There are many relevant producers defined in the OPF repository of method fragments. These include:

- **Business architect**, who rearchitects the business by creating new business models, processes, organizational structures, and recommends new applications.

- **Business strategist**, who leads the business strategy team, analyzes the customer organization, analyzes the market, and produces the business vision.

- **Customer representative**, who helps produce the business vision, provides requirements, and prioritizes the requirements from a business perspective.

- **Digital brand strategist**, who produces any requirements relating to digital branding.

- **Domain expert** (in the business domain, market, and application domain), who provides requirements, verifies the analysis, and provides recommendations regarding the business and application vision statements.

- **Process engineer**, who helps produce the requirements engineering process and evaluates the actual performance of the process.

- **Project manager**, who evaluates, approves, and manages the scope of the requirements.

- **Requirements engineer**, who leads the requirements team and performs most of the requirements tasks.

- **Security analyst**, who helps elicit, analyze and specify security requirements.

- **Software architect**, who helps prioritize requirements from an implementation perspective.

- **System architect**, who helps prioritize requirements from an implementation perspective.

- **Technical leader**, who evaluates and prioritizes the requirements from an implementation standpoint.

- **Technical writer**, who produces and maintains the requirements documents.

- **Technology strategist**, who analyzes the relevant technologies and their trends and helps establish any technology constraints.

- **Test engineer**, who ensures that all requirements can be validated (e.g., they are testable) and who acts as a liason to the various teams performing testing.

- **User analyst**, who analyzes users and user organizations.

- **User representative**, who helps analyze the user organizations and provides requirements from a user perspective.

Some teams for these producers are Business strategy team, Technology strategy team, Requirements team, Strategy inspection team, Requirements inspection team, Architecture team, Management team, and Quality team.

2.4 Construction and Usage Guidelines

There are several kinds of guidelines needed to engineer a project-specific method. These include, *inter alia,* method construction guidelines, tailoring guidelines and, less relevant here, extension guidelines – which assist the method engineer in modifying the metamodel itself. (Tailoring guidelines, which support minor modifications to the method once constructed also have less immediate impact on the topic of this paper.). Other important elements (not discussed further here) include sequencing rules, which can be expressed using pre- and post-conditions on (particularly) Tasks [14] and/or by ensuring the process and product perspectives are adequately connected [4].

A *construction guideline* helps method engineers both to instantiate (when necessary) the development process framework (metamodel) to create method components and also to select the best method components (from the Repository) in order to create the method itself [4,13,28,37]. Specifically, guidance is provided on the selection of appropriate work products, producers and work units as well as advising on how to allocate tasks and associated techniques to producers and how to group the tasks into workflows, activities etc. Finally, developmental stages (including phases and lifecycles) are chosen.

A commonly used, pragmatic approach to method construction is the following. Pick the work products you are willing to spend money on to create. Pick the appropriate tasks and techniques to produce them. Pick the appropriate producers to perform them, using tools where appropriate. Pick milestones and inch pebbles to schedule them and add to appropriate phases. Check for consistency. Document the method. Verify the method with stakeholders for acceptability and feasibility. Train teams in the method. Use the process. Iterate as appropriate. However, this needs experience. As an aid to helping the development team pick the best set of method fragments, the OPF suggests the use of a matrix [16] to describe this multi-faceted connexion between any pair of kinds of method fragment (e.g. Team and Task; Team and Role). Each matrix specifies the possibility value for each pair (e.g. each method component derived from the Team and Task metaclasses) either on a five-point scale [16] or, as here, as a binary value (Y/N).

3. Method Engineering Process

From a practical point of view, industrializing the above process of component selection involves an identification of the people involved. It is important that both management and development team representatives (the whole team if possible) be involved in creating the REM. Clearly, those involved need to have adequate skills that they can utilize in the selection of method components and their integration together. As well as project managers and requirements engineers, it may be beneficial to have an (internal or external) method engineer on the method engineering team.

Ralyté and Rolland [28] introduce a “Method Engineering Process Model (MEPM)” to construct the overall process with a complementary assembly process model (APM) for guiding the selection of appropriate method components; while Brinkkemper *et al.* [5] propose the use of a “method engineering language” that will assist in formalizing descriptions and usage of the various method components. Henderson-Sellers and Serour [18] propose a Trans-IT process which identifies method components specifically relevant to the introduction and inculcation of a software development process into an organization. Some of the more important elements of the method engineering process are discussed in the following subsections.

3.1 Determine Method Needs

The first step in producing an organization-specific requirements engineering method is for the members of the organizational method team to determine the goals and needs for their organizational method, of which requirements engineering is a critical part. These include robustness, repeatability, feasibility with respect to the organisational culture, measurable outcome, easy to learn, easy to follow, flexible. It is critical to understand the existing culture and the alternatives that may be perceived by both management and staff [32]. This study produces a documented statement of the specific needs of the organization and/or project team.

3.2 Learn OPEN Basics

The second step in producing an REM is for the members of the organizational method team to familiarize themselves with the OPEN Process Framework (OPF) in general and with the generic default OPF requirements engineering process framework (i.e., the subset of the OPF related to requirements engineering) in particular. They can begin by skimming the most recent OPEN books [13], skimming the overview webpages of the official OPEN website (http://www.open.org.au), and looking at the relevant webpages of the selected OPF tool (in this case, the OPEN Process Framework website[2], which provides over 1,000 free

[2] http://www.donald-firesmith.com

open source reusable OPF compliant method components). While doing this, the members of the method team should learn the concepts, terminology, and relationships between these concepts that are captured by the OPEN metamodel (e.g., what are the basic kinds of method components and how they relate to each other). They should also learn about the relevant reusable OPF method components (e.g., requirements engineering tasks, techniques, work products, roles). They should also familiarize themselves with the relevant OPEN method engineering tasks that they will be performing when producing their organizational-specific method-engineering framework for requirements engineering.

3.3 Select Method Components

The OPF contains numerous method components, not all are relevant to requirements engineering. Even the default requirements engineering subset of the OPF probably contains other related reusable method components that are not relevant to needs of any given organization, especially if that organization is limiting the type of endeavours it has in mind. The following steps should be followed for selecting the method components:

Step 1. Make a copy of the OPF generic default requirements engineering process framework as the initial draft version[3] of the organizational RE process framework.

Step 2. Use the output of the preceding *Determine Process Needs* task to decide which (if any) of the default method components in the organizational RE method component repository are either irrelevant or inappropriate to the needs of organization.

Step 3. Go through the list of currently available reusable method components, type-by-type, and component-by-component, and delete any method components that do not belong. The construction and usage guidelines (Section 2.4) may be helpful here.

Step 4. Just to be completely safe, also check the complete OPF to ensure that no useful cost effective method components were inadvertently left out when the OPF generic default RE process framework was created. This could be done using, for example, contingency factors (after [36])

Note that this is currently a manual task, the success of which will depend on the experience, skills, thoroughness and the care of the method engineers that perform it.

3.4 Extend the Process Framework Repository

The OPF repository is relatively large and complete, because it is based on the premise that it is easier to delete what you do not need from the repository than to add what you do need, especially if you are under typical project time and resource constraints. Nevertheless, it remains possible that the draft organizational RE method created thus far using only reusable method components from the OPF is incomplete. Now is the time to add any special method components that your organization might need in its organizational RE method. If

[3] http://www.donald-firesmith.com

these method components are not proprietary, you may also consider sharing them with the OPEN Consortium so that OPF can be updated with them and there will be fewer problems maintaining the organizational RE process framework if the OPF can be kept consistent with it.

3.5 Tailor the Method Components

Although the organizational REM now contains all of the appropriate method components, this does not mean that these method components are yet finished. They may need to be tailored in several ways. For example, the method component itself may be too large and complex for the needs of the organization. Excessive elements of these components may need to be removed. For example, the table of contents of documentation work products may contain sections that are inappropriate and should be removed. Similarly, the requirements team may contain too many roles, have too many objectives, or perform too many tasks.

3.6 Document the Method

Since the constructed method is to become the organization-specific or project-specific requirements method to be followed during RE, it is important that it is adequately documented and made available to *all* stakeholders (e.g., the members of the team as well as to all levels of management).

One way of ensuring that the documented method is maintained is to store it in some kind of database that can be used not only to generate the constructed method from its method components but also to undertake some checking for consistency. For example, it is important that any work product produced as part of the REM is either consumed in another part of the REM, delivered to other parts of the software development (e.g. design, test) or is delivered to the client.

3.7 Train the Staff

For the REM to be used successfully, all members of all teams must have buy-in and, potentially, “ownership” [32]. Training staff about the new method can often be one of the most risky components of this whole approach (see [14]) since the introduction of a new and innovative approach to an existing culture can often lead to resistance from the individuals concerned [2]. Staff need to be convinced that the introduction of the REM will be beneficial *to them personally* as well as to the organization for which they work. In addition, they must see that senior management are supplying sufficient resources and permitting them sufficient time to undertake the new learning experience [20,34].

Only when the development team members feel comfortable with their understanding of the new method is it appropriate to mandate the method [33]. Once such commitment has been gained from the development team, its use is likely to thrive. Without it, the project will surely fail [31].

3.8 Evaluation and Improvement of the Process

Process evaluation is a notoriously challenging research issue. Even success and failure are hard to define, being perceived differently by different parties [31].

For many organizations, it is not method adoption that it critical to their overall success but their success in creating a culture in which software process improvement (SPI) is the norm. To identify successes in SPI, it is common to utilize one of the existing capability assessment frameworks such as SPICE (Software Process Improvement and Capability dEtermination) [21] or the SEI's the Capability Maturity Model (CMM) [27] and, more recently, the CMMI [6].

In summary, then, the REM needs to be carefully maintained, sustained and improved as time progresses and the management and team members become increasingly adept in its use.

4. A Partial Example

Here we can give an example of how the above could produce an endeavour-specific RE method. Since we are creating a REM, then there is only a single method component that is relevant in the category of OPF's Activities viz. Requirements Engineering. Within that activity, many tasks need to be enacted by many teams, roles and people. Some possible linkages between teams and tasks[4] are given in Table 3 and between teams and roles in Table 4. For each team in the matrix, we list vertically all the possible Task (Table 3) or Roles (Table 4) and then ask whether the task/role is relevant to each team in turn (Y/N). This gives a first cut at the most appropriate method fragments to use and also identifies any unnecessary tasks/roles since these are indicated by a blank line in the final matrix. In these tables, we do not use the full five-value range discussed above, but merely a binary Y/N (blank means N). It is found from experience that this is adequate for the first adoption of an ME approach. With more experience, a more sophisticated use of the deontic matrices will become possibly, using all five deontic values.

Table 3 Deontic matrix to link RE Teams and Tasks

Associated Tasks	Team				
	Business strategy	Technology strategy	Requirements	Architecture	Customer representatives
Business visioning	Y				
Competitor analysis	Y				
Customer analysis	Y				
Market analysis	Y				
Requirements analysis	Y		Y	Y	Y

[4] Here we use the naming style of http://www.donald-firesmith.com rather than the style in the OPEN books in which OPF Tasks have imperative verb phrase names.

Requirements elicitation	Y		Y		
Requirements management	Y	Y	Y		
Requirements prototyping			Y		
Requirements specification	Y		Y		
Requirements reuse	Y	Y	Y		
Stakeholder profiling	Y				
Technology analysis		Y			
User analysis	Y				

Table 4 Deontic matrix to link Teams and their associated Roles

Associated Roles	Team A	Team B	Team C	Team D	Team E	Team F	Team G	Team H
Business architect	Y			Y				
Business strategist	Y			Y ind.	Y			
Customer representative	Y		Y	Y	Y			
Database architect					Y			
Digital brand strategist	Y			Y ind. (if relevant)				
Domain expert	Y		Y	Y ind.	Y			
Hardware architect					Y			
Metrics analyst					Y			
Method	Y			Y	Y			

engineer								
Project manager					Y			
Quality engineer				Y	Y			
Requirements engineer	Y		Y	Y	Y ind.			
Security analyst		Y	Y		Y			
Security architect					Y			
Software architect			Y		Y			
System architect			Y	Y	Y			
Technical leader				Y	Y			
Technical writer	Y	Y	Y	Y	Y			
Technology strategist		Y						
Test engineer			Y	Y	Y			
User analyst	Y				Y			
User representative	Y		Y					

Key to Teams: Team A = Business strategy team; Team B = Technology strategy team; Team C = Requirements team; Team D = Strategy inspection team; Team E = Requirements inspection team; Team F = Architecture team; Team G = Management team; Team H = Quality team.

ind. means Independent

Initially, the matrix is filled in from past experience: that of the process engineer, project manager, the team members and the external method engineer. As experience builds up, it becomes possible to create a database of past knowledge from which it is easier and more reliable to draw a first estimate of the likely linkages that will work *for that organizational context*. In time, it is anticipated that tools will be constructed to assist in this stage[26].

5. Discussion and Future Work

Davis and Zowghi [10] state that the purpose of requirements is to raise the likelihood that the right system will be built, i.e., that the system when built satisfies its intended customers and addresses their needs to an acceptable degree. It may be argued that the purpose is more short term, e.g., that its purpose is to ensure communication among all stakeholders, provide designers and testers with an oracle, guide project managers in allocation of resources, serve as a basis for requirements evolution and so on. However, Davis and Zowghi argue that each of these short term objectives are important only because of their strong correlation to the real purpose of requirements, i.e., to raise the likelihood that the right system will be built. They further state that a "*good" requirements practice* is one that either reduces the cost of the development project or increases the quality of the resulting product when used in specific situations. Few requirements practices have been validated as "good" in practice, and those that have, rarely if ever, describe the specific situations where they are effective. However, we do have a variety of sources of requirements practices for which the authors do claim goodness. For example, Sommerville and Sawyer's [35] *Good* Practice Guide, Weigers [38] provides a chapter on *good* practices, and the Robertsons' book [30] discusses *Mastering* the requirements process.

Following construction of the REM and its utilization on a project, it is important to follow up and enquire about its effectiveness in practice Some early results on exploring the effectiveness of OPF are reported in a series of papers including [31-33] from two industry case studies. While these papers focussed on the success or failure indicators, primarily in terms of people, culture and organization, there are other questions that could be asked in any future evaluation survey. These include:

How big was the job? (person days)

How easy/hard was it to use OPEN?

How quick was it to construct the tailored process – was it overly labour intensive?

Which parts could benefit from automation?

Was the method component repository complete?

Were method components adequate?

Were the method components adequately documented?

Can you evaluate the quality of the method component repository?

Can you evaluate the quality of the ensuing REM?

Did the approach permit or support process improvement?

Has the ME approach turned out to be cost effective?

Is an ME approach practical in an industry setting?

We plan to undertake such surveys with companies adopting the OPF- and ME-based approach to requirements engineering. The results of these surveys will be the subject of a later paper.

6. Conclusion

Since it is not possible to identify or to create a single method that is appropriate for all situations, the need for an REM necessitates the search for a mechanism that will support the flexible creation of a number of tailored REMs from a single base – here a repository of method components, based on that of the OPEN Process Framework and the techniques of method engineering, is used to illustrate how a project-specific REMs can be generated, applied and maintained.

References

1. Boehm, B.W. and Papaccio, P.N., 1988, Understanding and controlling software costs, *IEEE Trans of Software Engineering,* **14(10),** 1462-1477

2. Bridges, W., 1995, *Managing Transitions, Making the most of change*, Nicholas Brealey Publishing, UK.

3. Brinkkemper, S., 1996, Method engineering: engineering of information systems development methods and tools, *Inf. Software Technol.*, **38(4)**, 275-280.

4. Brinkkemper, S., Saeki, M. and Harmsen, F., 1998, Assembly techniques for method engineering, Procs. CAiSE'98, LNCS 1413, *Advanced Information Systems Engineering* (ed. B. Pernici and C. Thanos), Springer-Verlag, Berlin, 381-400

5. Brinkkemper, S., Saeki, M. and Harmsen, F., 2001, A method engineering language for the description of systems development methods (extended abstract), *CAiSE 2001* (eds. K.R. Dittrich, A. Geppert and M.C. Norrie), LNCS 2068, Springer-Verlag, Berlin, 473-476

6. CMU, 2002, Capability Maturity Model Integration (CMMI-SE/SW/IPPD/SS) Version 1.1, *CMU/SEI-2002-TR-011 and CMU/SEI-2002-TR-012*

7. Cockburn, A., 2000, Selecting a project's methodology, *IEEE Software*, **17(4)**, 64-71

8. Curtis, B., Kellner, M.I. and Over, J., 1992, Process modelling, *Comm. ACM*, **35(9)**, 75-90

9. Davis, A.M., 1993, *Software Requirements: Analysis and Specification,* Prentice Hall, second Edition, 1993.

10. Davis A.M., Zowghi D., 2005,"Good requirements practices are neither necessary nor sufficient ", Viewpoints, Requirements Engineering Journal, Vol 10. available online 8th October 2004

11. Firesmith, D.G., 2002a, Requirements engineering, *J. Object Technology,* 1**(4)**, 93-103

12. Firesmith, D.G., 2002b, Requirements engineering, *J. Object Technology*, 1**(5)**, 83-94

13. Firesmith, D.G. and Henderson-Sellers, B., 2002*, The OPEN Process Framework. An Introduction*, Addison-Wesley

14. Goldberg, A. and Rubin, K., 1995, *Succeeding with Objects*, Addison-Wesley

15. Graham, I., 1995, A non-procedural process model for object-oriented software development, *Report on Object Analysis and Design,* **1(5)**, 10-11

16. Graham, I., Henderson-Sellers, B., and Younessi, H., 1997*, The OPEN Process Specification*, Addison-Wesley, 314pp

17. Henderson-Sellers, B., 2003, Method engineering for OO system development, *Comm. ACM,* **46(10)**, 73-78

18. Henderson-Sellers, B. and Serour, M., 2000, Creating a process for transitioning to object technology, *Proceedings Seventh Asia--Pacific Software Engineering Conference. APSEC 2000,* IEEE Computer Society Press, Los Alamitos, CA, USA, 436-440

19. Hruby, P., 2000, Designing customizable methodologies, *JOOP*, **December 2000**, 22-31

20. Humphrey, W.S., 2000, *Introduction to the Team Software Process*, Addison Wesley

21. ISO/IEC, 1998, TR15504, Information technology – software process assessment, in 9 parts, International Standards Organization, Geneva, Switzerland

22. Jirotka, M. and Goguen, J., 1994, Requirements Engineering social and technical issues, Academic Press

23. Kotonya, G. and Sommerville, I., 1997, Requirements Engineering processes and techniques, Wiley

24. Kumar, K. and Welke, R.J., 1992, Methodology engineering: a proposal for situation-specific methodology construction, in *Challenges and Strategies for Research in Systems Development* (eds. W.W. Cotterman and J.A. Senn), J. Wiley, Chichester, 257-269

25. Macaulay, L., 1996, *Requirements Engineering*, Springer

26. Nguyen, V.P. and Henderson-Sellers, B., 2003, Towards automated support for method engineering with the OPEN Process Framework}, *Procs. Seventh IASTED International Conference on Software Engineering and Applications* (ed. M.H Hamza), ACTA Press, Anaheim, CA, USA, 691-696

27. Paulk, C., Weber, C.V., Garcia, S., Chrissis, M.B. and Bush, M., 1993, Key Practices of the Capability Maturity Model, Version 1.1, CMU/SEI?93?TR?25, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA

28. Ralyté, J. and Rolland, C., 2001a, An assembly process model for method engineering, in K.R. Dittrich, A. Geppert and M.C. Norrie (Eds.) *Advanced Information Systems Engineering*), LNCS2068, Springer, Berlin, 267-283.

29. Ralyté, J. and Rolland, C., 2001b, An approach for method engineering, *Procs. 20th Int. Conf on Conceptual Modelling (ER2001)*, LNCS 2224, Springer-Verlag, Berlin, 471-484

30. Robertson, J., and S. Robertson, 1999, Mastering the Requirements Process, Addison Wesley

31. Serour, M.K. and Henderson-Sellers, B., 2002, The role of organizational culture on the adoption and diffusion of software engineering process: an empirical study, *The Adoption and Diffusion of IT in an Environment of Critical Change* (eds. D. Bunker, D. Wilson and S. Elliot), IFIP/Pearson, Frenchs Forest, Australia, 76-88

32. Serour, M.K. and Henderson-Sellers, B., 2004, Introducing agility: a case study of situational method engineering using the OPEN Process Framework, *Procs. 28th Annual International Computer Software and Applications Conference. COMPSAC 2004*, IEEE Computer Society Press, Los Alamitos, CA, USA, 50-57

33. Serour, M., Henderson-Sellers, B., Hughes, J., Winder, D. and Chow, L., 2002, Organizational transition to object technology: theory and practice, *Object-Oriented Information Systems* (eds. Z. Bellahsène, D. Patel and C. Rolland), LNCS 2425, Springer-Verlag, Berlin, 229-241

34. Serour, M.K., Dagher, L., Prior, J. and Henderson-Sellers, B., 2004, OPEN for agility: an action research study of introducing method engineering into a government sector, *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education* (eds. O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic and S. Wrycza), Vilnius Gediminas Technical University, Vilnius, Lithuania, 105-116

35. Sommerville I. and Sawyer P., 1997, Requirements Engineering, a good practice guide, Wiley

36. ter Hofstede, A.H.M. and T.F. Verhoef, 1997. On the feasibility of situational method engineering. *Information Systems.* **22**(6/7), 401-422

37. van Slooten, K., Hodes, B., 1996, Characterizing IS development projects, **in** *Procs. IFIP TC8 Working Conf. on Method Engineering: Principles of method construction and tool support* (eds. S. Brinkkemper, K. Lyytinen, R. Welke) Chapman&Hall, Great Britain, 29-44

38. Wiegers, K., 2003, Software Requirements, Microsoft Press