Object Oriented Hybrid Software Engineering Process (SEP) model for Small Scale Software Development Firms

Shaikh Mostafa Al Masum
mostafa@miv.t.u-tokyo.ac.jp
Research Student,
Ishizuka Lab, University of Tokyo, Japan

A. S. M. Mahbub Morshed
mdmorshed@hotmail.com
Student, Islamic University of Technology (IUT), Board Bazar, Gazipur 1704, Bangladesh

Ishizuka Mitsuru
ishizuka@miv.t.u-tokyo.ac.jp
Professor, Department of Information and Communication Engineering, University of Tokyo, Japan

ABSTRACT*: Software Engineering Process (SEP) is a time sequenced set of activities to transform users' requirements into a software. There are many SEP and methodologies, namely Rational Unified Process (RUP), Object-Oriented Process, Environment, and Notation (OPEN), Extreme Programming (XP), etc, having support for different scales of development. All these methodologies are mostly Object Oriented and tailored for mid to large scale of development. In most of the cases these process models are very elaborative that leads to the necessity of having mentors to configure a specific process. This paper actually distinguishes some characteristics of software development activities of Small Scale Software Development Firm (SSSDF) and hence, proposes a preconfigured hybrid model of SEP in the light of few third generation methodologies. Firstly, this paper provides a brief comparison of the existing third generation methodologies. Then it depicts some development characteristics of SSSDF which are actually revealed from survey of 15 BSDF. And finally a hybrid model of SEP is realized.*

Keywords*: Hybrid Software Engineering Process, OPEN, RUP, XP, Small Scale Software Development (SSSD), Software Development Life Cycle (SDLC).*

1. INTRODUCTION

Like all other engineering processes, SEP also follows specific methodologies, provides the guidelines that suggest some specific tasks or activities to be performed at different stages of SDLC to build any software. At present, third generation methodologies like RUP[1], OPEN[2], XP[4][8], etc. are used for developing software. In the next section a brief summary of three popular third generation OO methodologies is depicted.

2. OPEN : AN OVERVIEW

Object-oriented **P**rocess, **E**nvironment, and **N**otation (OPEN) is a full-life cycle, object-oriented software development approach.

Open Process Framework (OPF) contains 5 groups of components namely,

1. *Work Products* – components that are developed by the project.
2. *Languages* – the medium used to document a work product.
3. *Producers* – anything that produces a work product.
4. *Work Units* – a set of cohesive operations performed by the producer to build a work product.
5. *Stages* – the time intervals that provide a macro organization to the work units.

The figures below show how the different components of OPF are related with each other.

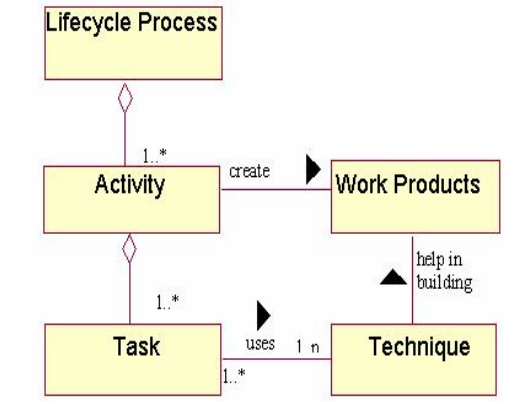

Figure 1: Metamodel of OPEN (after Henderson-Sellers)

A process is instantiated from the OPEN process metamodel and then tailored by adding or subtracting process components such as activities, tasks, techniques in order to best fit the organizational needs in terms of the organization's size, culture, investment and other characteristics. OPEN process lifecycle has a set of activities to produce specific work products by stepwise tasks and finally techniques explain about the procedure to produce specific work products.

3. RUP : AN OVERVIEW

Rational **U**nified **P**rocess (RUP) is a heavyweight object-oriented software development process. It emphasizes the adoption of certain *best practices* of modern software development. The RUP weaves those best practices into the definitions of following terms.

- *Roles* - sets of activities performed and artifacts owned
- *Disciplines* - focus areas of software engineering effort such as *Requirements*, *Analysis and Design*, *Implementation*, and *Test*
- *Activities* - definitions of the way artifacts are produced and evaluated
- *Artifacts* - the work products used, produced or modified in the performance of activities.

The RUP is an iterative process that identifies four phases of any software development project. Over time, the project goes through Inception, Elaboration, Construction, and Transition phases.

4. XP: AN OVERVIEW

E**x**treme **P**rogramming (XP) is a lightweight object-oriented software development process developed by Kent Beck in 1996 and it is based on four values namely,

1. *Communication* – XP programmer communicate with their fellow programmers and with customers by having an on-site customer throughout the development lifecycle.
2. *Simplicity* – XP programmer keep their design simple and clean by removing duplication and complexity from codes and by maintaining minimum number of non-code related artifacts.
3. *Feedback* – XP programmer get feedback by testing their soft ware starting from day one.
4. *Courage* – XP programmer need to be honest about what they can and cannot do. They should courageously respond to changing requirements and technology even if that means breaking away from the current trend.

The following twelve XP practices support the four values. They are the planning game, Small releases, Metaphor, Simple design, Testing, Refactoring, Pair programming, Collective ownership, Continuous integration, Forty-hour week, On-site customer and Coding standards. These practices are quite self explanatory and are not discussed here in this scope.

5. COMPARISON OF METHODOLOGIES

The ideal approach to comparing any three processes is to evaluate three processes in practice[3]. Unfortunately, such a comparison experiment is extremely difficult to undertake because of the inability of an empirical software engineering researcher to control the many confounding variables or to replicate the experiments (Menzies and Haynes, 1994).

Thus in the evaluation presented here, the focus is on a "theoretical" comparison between OPEN, RUP and XP. It should be noted that the comparison is not quite exhaustive. Only those aspects are presented which will help in developing the hybrid model.

5.1 Meta-model and Flexibility

OPEN is defined at the meta-model level whereas both RUP and XP are defined at the model level albeit their models were instantiated from meta-models somewhat similar to that of OPEN.

Organization-specific processes are instantiated from the OPEN meta-model by choosing specific Activities, Tasks and Techniques (three of the major meta-level classes) and specific configurations thereof. Process tailoring may also be needed whereby details of the Tasks and Techniques are "tweaked" for optimum fit to the problem domain. This makes OPEN processes very flexible. RUP and XP, on the other hand, supports comparatively lesser flexibility because both of them are pre-packaged, pre-configured instance (i.e. Hruby, 2000) of their own meta-model and could thus be described as a tailored methodology. So, the developer does not have to and can not regenerate a process, it is already available. Some tailoring is possible in RUP and no such possibility is there in XP.

5.2 Time and Effort Allocation

This part discusses how each process is arranged over time and how the staffing effort allocation compares.

The OPEN life span is divided into 6 phases – Business Modeling, Inception, Construction, Usage, Retirement and Business Re-engineering. Each phase may go through a number of builds that last between one to three months. The entire duration is usually more than 2 years.

The life span of RUP is divided into 4 phases – Inception, Elaboration, Construction and Transition. Each phase is a summation of some – usually 3 to 9 – iterations that can last between 2 weeks and 6 months. The expected project duration is from 6 weeks to 54 months.

The entire lifespan of XP is considered as 1 large phase and iteration take place over the entire lifespan. The duration of iteration is about 2 weeks. The duration of the project is about 2 months approximately.

5.3 Artifacts

Artifacts – or work products for OPEN - are any components or deliverables used, produced or modified by the project. Artifacts include user-manual, use-cases, test fixtures, project plan, etc.
Both RUP and OPEN describes a large number of artifacts – RUP describes over 100 of them – that have to be produced over the entire SDLC for a complete software. These artifacts capture the results of various activities. XP also tries to capture the results but provides little guidance on how to do it. So, XP has around 30 artifacts. In XP, the final resting place for requirements or design decisions is the code, not artifacts. Unfortunately, code is not an effective communication medium for all stakeholders.

5.4 Activities

An activity – a task for OPEN – is a major work unit that produces a related set of work products. Activities describe what needs to be done, not how. Generally, the more the activities, the more time it will take to build the project.

In OPEN, there are 7 major activities and many more sub-activities. The major activities are Project initiation, Requirements engineering, Analysis and model refinement, Project planning, Build, Evaluation and Implementation Planning (or Deployment). RUP has 9 activities – Business Modeling, Requirements, Analysis and Design, Implementation, Test, Deployment, Configuration & Change Management and Project Management. Lastly, XP has only 4 activities - Coding, Testing, Listening, And Designing.

The activities show that XP has a very simple view of software development as opposed to OPEN that has a strong focus not only on software development but also on project management (Henderson-Sellers and Due, 1997), business decision making, sociological context and integrated reuse.

5.5 SDLC Model

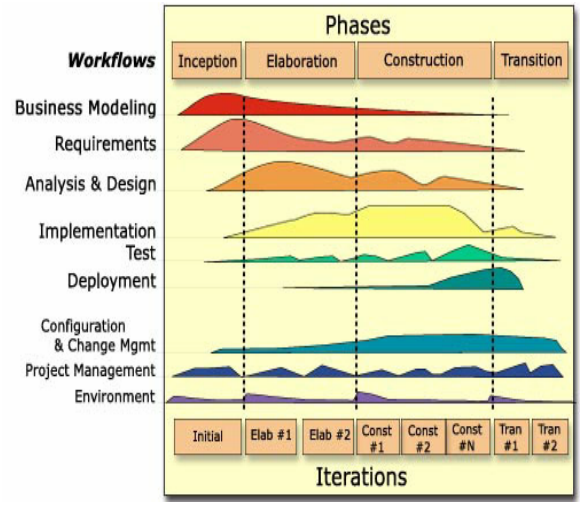

Figure 2: The phases and iterations of RUP

The SDLC model of XP is similar to that of RUP except that there is fewer number of iteration at each phase and it has only 4 workflows – Design, Code, Test and Listen – on the vertical axes.

There are many ways in which XP is similar to RUP and both can be instantiated from OPEN's meta-model. XP can be understood as a shorter version of RUP without having the additional features of RUP like project inception, deployment, business modeling, etc. However, XP is not particularly suitable for those big projects which require extra planning, iteration and requirement engineering. In contrast, OPEN can be used for all projects provided that a robust process is instantiated from the meta-model.

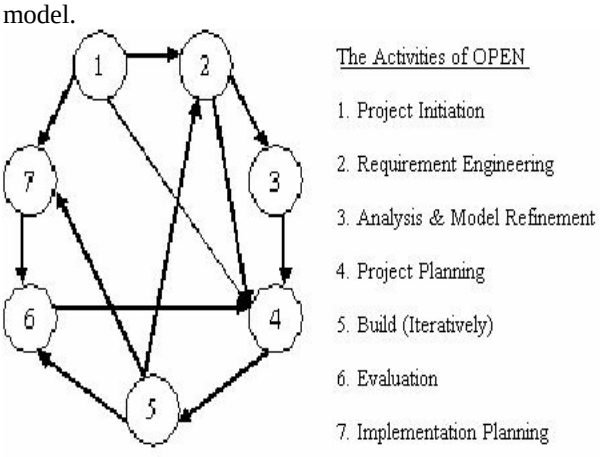

Figure 3
A typical SDLC instantiated from OPEN metamodel

6. FEATURES OF SSSDF

Some of the development features of SSSDF based on the results of a survey of 15 leading Bangladesh Based (both Domestic and International) Software Firms is listed below:

a. Firms are small – average number of developers is 25.
b. Web based solution is the major type of development work.
c. Almost all firms use Object Oriented Approach.
d. Most of them don't follow any well-known methodology, but some try to follow some variant of RUP.
e. All works are done in a project driven small team – average team size 5/6 programmers.
f. Average project duration is 8 to 10 months.

Some criticisms regarding the existing OO Methodologies with respect to these points are made. OPEN is not purely ideal for SSSDF because of its "mammoth-like" heavy-weight nature and necessity of high degree of expertise by the developers. OPEN is highly flexible meta-level framework, together with a repository of process component instances, from which industry creates their own organizationally tailored method. OPEN gives a high degree of flexibility although care must be taken in optimizing this construction process and the resultant OPEN process instance. Since, at present, this task needs to be undertaken by a skilled process engineer (Henderson-Sellers *et al*., 2001); it would not be ideal for SSSD environment which usually lacks highly professional and expert software developers.

As already mentioned RUP and XP, on the other hand – are pre-packaged, pre-configured instances (i.e. Hruby, 2000) of their own meta-model and could thus be described as a (tailorable) methodology. So, the developer does not have to regenerate a process due to its readiness. Now, project managers' task would be to determine which of these two processes would best suit SSSDF.

RUP also has to bear some brunt of criticism faced by OPEN because pre-configured and pre-packaged instances of RUP can also accommodate a variety of processes, albeit the varieties are not as numerous as available in OPEN. But, unlike OPEN, RUP can not be totally thrown away as a possible candidate methodology for SSSDF due to its remarkable adaptability and, most important of all, it already has got some followers in SSSDF.

XP has become very popular recently all over the world due to its simplistic nature and several practices. Since it is relatively latest methodology, it enjoys a large number of comparative advantages over its counterparts. So, it is most appropriate to first investigate whether XP can blend and be well applicable to SSSDF for all sorts of development works.

XP - in its orthodox form - addresses only a narrow range of software development projects (mainly small projects) – i.e. XP can not be used for all projects. The following conditions have been identified and XP's practices can be fully implemented to develop an "extreme" project given that all the following conditions are fulfilled,

a. The project team must be small – ideally ten people or less.
b. The project itself has to be small – the one that can be completed within 2 months.
c. The team must be co-located, and willing and able to do pair programming.
d. There must be an on-site customer during the whole duration of the project.

The XP's pair-programming practice can also be easily implemented in SSSDF as all the team members are co-located within the firm to produce robust code and increase reusability. However, care must be taken that the paired team members have compatible personalities and well-matched programming skills.

Originally, XP admits that a real customer must sit with the team, available to answer questions, resolve disputes, and set small-scale priorities. Commitment of an on-site customer is not particularly forthcoming as customers may not be free for the duration required by a XP process. On this regard, however, the RUP is more flexible. The RUP acknowledges that it is not necessary to have a real customer co-located with the development team.

Even though XP seems to be more suitable for SSSDF but there are some aspects that are not covered by XP but are crucial for SSSDF which are not trivial:

a. In XP, only unit and acceptance tests are applied on the system and the development team uses the test results to decide whether the system is ready for the customer. Since most SSSDFs' works involve web-based solutions, other tests may be required: for example, load

tests for Web sites whereas these extra tests are available in RUP and OPEN.

b. The whole area of system deployment is missing from XP. Like all Commercial software products, most Small Scale Software firms also require online documentation, packaging, distribution, user manuals, training materials, and a support organization. The RUP Deployment discipline provides the guidance to practitioners on how to create appropriate materials and then use them.

c. XP doesn't encourage for UML like diagrams which are very essentials for SSSDF where developers often leaves jobs.

As it can be seen, XP – on its own – cannot fulfill the development requirements of SSSDF. RUP, on the over hand, does an overkill and have too many additional aspects which are not actually required at present for SSSDF. So, a suitable methodology for SSSDF is neither XP nor RUP but rather, should contain a mixture of characteristics taken from both XP and RUP and can be instantiated from OPEN's meta-model to make software development more predictive, managed and streamlined.

7. THE HYBRID METHODOLOGY

The proposed hybrid (figure 4) model has four phases: Inception, Planning, Iteration and Deployment.

The main goal of the inception phase is to determine the true objectives of the user and to devise user stories for the planning phase. A working version of the software is developed in each iteration. However, the number of iterations can be much less than that of RUP. Some tests, library management and manuals are developed in the deployment phase along with the delivery of the software. The test follows iterative waterfall model and is given the same kind of emphasis as that of XP.

During Project Initiation we restrict three activities only. These are Feasibility Study, Cost-Benefit Analysis and Identification of the Reusability Factor. Actually these three activities are very essential for the small firms to make a decision regarding project induction. We feel that collection of "User Story" is important to shape the project. The same procedure followed in XP can be followed. We strictly suggest for UML like diagrams to make Analysis and Business Model to minimize the risk that is most likely to encounter by sudden leave of software developers in SSSDF and which is, in fact, very common. According to the verified and accepted software business model (illustrated by diagrams), a Release plan is made. .

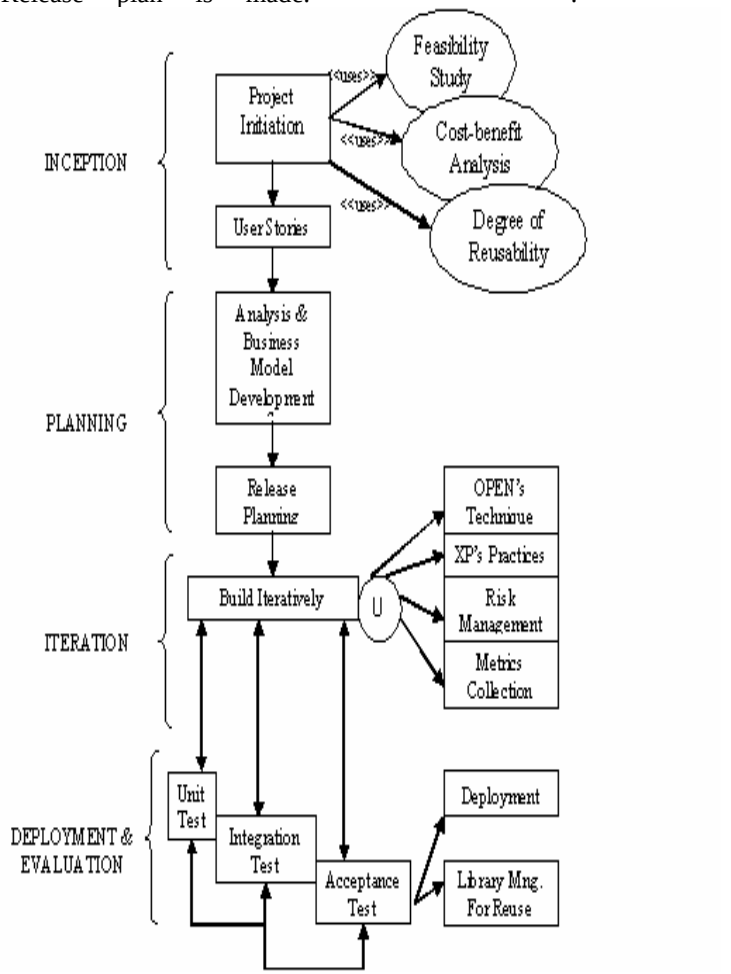

Figure 4: Object Oriented Hybrid Methodology

According to release plan developers are divided into small units within the project to write the codes and build the system. Building is strictly related with testing which must follow a water-fall model for testing. During Iteration Phase developers build iteratively with the help of some OPEN's OO Techniques [7] and follow XP's practices[4][8]. In this phase Risk Management and Metric Collection is suggested through waterfall model of testing strategies. After acceptance test Deployment and Library Management for reusable components are considered.

8. CONCLUSION

Small-scale software development is becoming popular day-by-day and this trend is expected to continue and flourish in future. OPEN and RUP mainly targets the projects of long duration, higher complexity and managing a large number of people. On the other hand, XP has some deficits regarding some valuable process components. Hence, we believe that the proposed model, tuned for small-scale software, will become more applicable and easily bearable to the emerging small scale software firms now and for the days to come because it has incorporated several good practices from XP fitting with RUP like phases abiding by a meta-model similar to OPEN. We expect the proposed model will be able to streamline that ad-hoc software development process involved in Small Scale Software Development Firms.

REFERENCES

[1] Booch, Grady et al, *The Unified Modeling Language – User Guide,* Addison-Wesley 2000.

[2] Henderson-Sellers, B. and Due, R.T., *OPEN project management*, Object Expert, 2(2), 30-35 1997, invited article for Object Expert, 2(2), 30-35 COTAR Contribution no 96/10

[3] Hruby, P., 2000, "Designing customizable methodologies", *JOOP*, 13(8), 22-31

[4] Kent Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley 1999.

[5] Menzies, T. and Haynes, P., "The methodology of methodologies; or, evaluating current methodologies; why and how", *Proceedings of the Technology of Object Oriented Languages and Systems Pacific conference: Prentice Hall, 1995.*

[6] Ron Jeffries et al, *Extreme Programming Installed*. Addison-Wesley, 2000.

[7] Henderson-Sellers, B., Simons, A., Younessi, H., *The OPEN Toolbox of Techniques*, Addison-Wesley Pub Co; Book and CD-ROM edition (June 17, 1999)

[8] http://extremeprogramming.org, Official Website of Extreme Programming Community.