Total Software Process Model Evolution in EPOS
Experience Report

Minh N. Nguyen, Alf Inge Wang and Reidar Conradi
Div. of Computer Science
Norwegian University of Science and Technology (NTNU)
N-7034 Trondheim, Norway
Phone: +47 73594485, Fax: +47 73594466
Email: {nguyen,alfw,conradi}@idi.ntnu.no

ABSTRACT
This paper presents a case study of Norwegian banking soft-
ware house where the objective is to adopt a categorization
framework for managing evolution in software projects to
identify project profiles and evolution patterns, and to suggest
improvements to better support frequent evolutions. Based
on an analysis of collected evolution data from an ongoing
case study, we elaborate a QIP-inspired method and own
techniques to evolve corresponding process models in our
Process-centered Software Engineering Environment, called
EPOS. The method describes also how to synthesize and
reuse evolution experience from completed projects to im-
prove planning and estimation in new similar projects. The
collected data demonstrates that requirement changes which
are detected in later development phases, are major causes for
cost overruns in the studied organization.

Keywords:
Process model evolution, experience reuse and learning, cat-
egorization framework for process evolution, evolution pat-
tern, empirical evolution data.

INTRODUCTION
The software industry needs to develop high quality software
predictably on time and budget. Much research has there-
fore focused on technologies for Software Process Improve-
ment (SPI), i.e. techniques for improving the productivity and
quality of associated processes. We can mention efforts such
as SEI-CMM [PWCC95], OIP/GOM [BCR94a], Bootstrap
[HMK+94] and SPICE [Dor93], etc. Likewise, many case
studies have been conducted in software organizations to val-
idate the applicability of proposed technologies [BCM+92]
[PC94]. However, there is insufficient research addressing
the innumerable and unforeseen process changes that occur
during normal software projects. Frequent changes and in-
terruptions are considered as major cause for late delivery,
cost overrun, missing features, and thus poor quality. The
ability to handle unexpected events occurring both within
the organization and in the surrounding environment is thus
claimed to be a characteristic feature of successful compa-
nies. We need to improve our ability to predict, plan and man-
age changes based upon previous experiences. Adequate en-
actment support for process changes, embedded in a PSEE,
is also considered necessary and desirable.

Our previous work [NC94] has been revised to contain a
classification of process change and their impact on the en-
acting process (project) in the form of recognizable evolution
patterns. Such patterns associated with typical project
and product processes will contribute to building an empirical
base. By utilizing this base the predictability in project
planning/scheduling process is improved and more confident.
We have collected empirical evolution data from several soft-
ware projects to provide us better understanding of the actual
evolution profile. Appropriate support has then been intro-
duced in our PSEE to better manage the observed evolution.

RELATED WORK
Some research has been dedicated to process model evolu-
tion. In [JC93] and [BFG93], basic mechanisms and tech-
niques for process model evolution are identified and imple-
mented in two different PSEEs, EPOS and SPADE respec-
tively. These efforts have emphasized methods and mech-
nisms for changing process model fragments, represented
as types or classes in a versioned repository. No concrete
couplings have been made with actual evolution in real soft-
ware projects. On the other hand, Madhavji provided a
methodological perspective of evolution [Mad91] and envi-
ronmental facilities (e.g., dependency and change structure)
for changes in the Prism model [Mad92]. The model focuses
on managing consistent change propagation in a feedback-
based environment. Unfortunately, little effort has been dedi-
cated to pursuit and apply the work which still remains at
model level. The major focus in [LBSS] and [Leh94j is on
classification and studies of evolving entities, especially soft-
ware systems, referred to as Program Evolution Dynamics.
In this work, Lehman has identified five evolving entities in
the software process: revision/version, S-type program1, F-
type application2, process, and process model. However, this
work does not offer sufficient detail in systematizing the im-
pacts of evolution patterns in a concrete context. Work by
DeMarco [DeM82], in the TAME project [OB92] and empir-
ical studies at NASA-SEL [MPP+94] discuss how to achieve
control and to improve estimation accuracy in rather large
software projects. However, these results are not necessarily

1 S-type program is a program which satisfies the fixed initial
specifications.
2 E-type application is subjected to continuous evolution as environmen-
tal conditions change.
applicable for small and medium enterprises (SMEs). Large software projects are far more complex due to number of persons, number of software components and required amount of management involved. In general, there is little progress in obtaining and keeping control over evolving software processes by exploiting technologies from software measurement [Fen91] or from experience reuse/synthesis [DCR94b] [DR91]. Indeed, most project management tools are of limited practical value, due to the accumulated effect of process changes during project execution. Thus, EPOS attempts to integrate process and project support in managing changes during development.

CONCEPTUAL AND CATEGORIZATION FRAMEWORK

We distinguish between a changing real world where managerial and technical activities take place, and a modeled world, where the human perceptions and constraints of the real world is represented by models and documentation. The former world is continuously evolving due to changing needs and perceptions. The latter model remains static, until humans determine to change it according to the world it reflects.

A software process is a set of managerial and technical activities applying certain technologies (methods and tools) to transform a requirement into a software system. The process consists of three parts: a production process (p-p), a meta-process (m-p), and process support (p-s). The primary goal of the p-p is to develop a software system in a project context with limited resources (humans, production tools) and budget (scheduled cost and time). Furthermore, the p-p should adhere to a plan acting as a project-specific process model. The m-p describes and governs overall planning and executing of p-p, as well as packaging of gained experiences and evolving the entire software process. The p-s is comprised of a model of the real process (p-p and m-p) expressed in process modeling language(s) and manipulated by a set of process tools (a PSEE).

There are two types of human agents interacting with and continuously raising change pressure to a software process. External agents consist of suppliers/sub-contractors and competitors. Internal agents comprise senior people (executive managers); middle (line and project managers); process (QA, process designer); SW engineers (analysts, designers, developers, testers, service people, etc.), and customer representatives. External factors, such as market trends/forces, technology availability, and unanticipated delays/distortions, also influence and drive process evolution.

Below, we present a categorization framework for process evolution. A detailed description can be found in [NC96]. Our framework distinguishes between where, why, what, when, how and by-whom process changes are introduced. Those six dimensions can be hierarchically decomposed for refinement and future extensions.

Where: identifies the sources that request or cause a given process change.
Why: represents the major causes (drivers) behind changes. They are used for causal analysis.
What: describes what process parts (p-p, p-s, m-p) are requested to be changed or affected by this action.
When: distinguishes between the time when the change request is detected (Change Detection Time - CDT) and the time when the proposed change is designed and implemented (Change Realization Time - CRT).
How: records the corrective and preventive actions being conducted to handle a given process change. It contains organizational change, technological innovation or plan adjustment.
By-whom: identifies the human agents who approve and perform the change action.

Each dimension can be further decomposed into several aspects. Each aspect is represented by appropriated categories. An observed process evolution, categorized by the proposed framework, is called an evolution pattern. The evolution pattern is instrumented by a cost measure in term of gain or loss of productivity or progress. Figure 1 depicts the elaborated categorization framework for process evolution which is used in the case study.

Figure 1: Categorization framework for process evolution

Where: identifies the sources that request or cause a given process change.
Why: represents the major causes (drivers) behind changes. They are used for causal analysis.
What: describes what process parts (p-p, p-s, m-p) are requested to be changed or affected by this action.
When: distinguishes between the time when the change request is detected (Change Detection Time - CDT) and the time when the proposed change is designed and implemented (Change Realization Time - CRT).
How: records the corrective and preventive actions being conducted to handle a given process change. It contains organizational change, technological innovation or plan adjustment.
By-whom: identifies the human agents who approve and perform the change action.

Each dimension can be further decomposed into several aspects. Each aspect is represented by appropriated categories. An observed process evolution, categorized by the proposed framework, is called an evolution pattern. The evolution pattern is instrumented by a cost measure in term of gain or loss of productivity or progress. Figure 1 depicts the elaborated categorization framework for process evolution which is used in the case study.
EPOS
This section gives you an overview of the PSEE EPOS and how EPOS support process evolution through mechanisms and tools.

EPOS System Overview
EPOS [MCL+95] [C+94] is a software process modeling and enactment system. EPOS supports a reflexive, object-oriented software process modeling language called SPELL [C+92]. Several different sub-models are supported in EPOS for describing activities and products. These sub-models are: Activity (task) model, Product model, Tool model, Human and Role model, Cooperation model and Meta-Process model.

To support process modeling and evolution, we facilitate basic mechanisms for incremental (re)planning and enactment of the process models by process tools like the Planner and Process Engine [JC93] [LC93]. We have built the EPOSDB [Mun93] to store versioned software products, as well as their related process models. EPOS also supports cooperative transactions.

Mechanisms for managing process evolution in EPOS
To manage evolution of a software process, it is necessary to store and utilize information characterizing a project and information about changes that are made to the project. It is also important to have PSEE-tool support for changes made to the process model and instantiated process representation. The two next subsections will describe experience database support in EPOS and how EPOS support process model manipulation. The three following subsections describe how to change the instantiated process representation on the fly.

Retrieval of project experience
Project experience is stored in the Experience database as an evolution pattern and associated with a particular project and product profile. Two characterization forms have been made to retrieve project and product profiles of the new project. Each characteristic is instrumented by a corporate-specific weight. This weight indicates the importance or dominance of a given characteristic when the degree of similarity is determined during selection process of baseline project. Information from the filled-out forms is used to select candidates among completed projects from the database. The baseline project is chosen by summarizing the weights of matching characteristics. The evolution data of the baseline project is available for planning and comparison.
**Recording of project experience**

While executing a project, changes are recorded by filling out an *evolution request* form. This form collects data sufficient to categorize a given change into an evolution pattern which is stored back to the Experience database with its impact in term of cost. The project performance and product quality measures are also recorded for future learning. On the other hand, new experiences may lead to a revised process model and revised corporate profiles. Such support is provided by the Schema Manager tool which is a tool for changing the process model.

**Manipulation of task-network layout**

The following techniques can be used to edit the task-network (making changes to the instantiated process representation during enactment):

- **Delete-task**: Remove an task from the task network. This operation can only be done if neighbor tasks can be coupled together.

- **Add-task**: Input an additional task to the task network. This operation can not be applied to the task-network, if it leads to an inconsistent task-network.

By using combinations of the two above you get:

- **Split-task**: Replace one task with two or more new tasks placed side by side. The new tasks are a subset of the task that was replaced.

- **Merge-task**: Replace two or more tasks by one new task. The new task is a collection of all tasks replaced.

**Manipulation of task-network scheduling**

It is sometimes required to modify the schedule of the task-network to adapt to various change incidents. Such incidents vary from need to revise initial effort estimates to a re-execute or to postpone a start date of a particular task. During execution on a task-network, we can change individual properties of tasks, such as:

- **Start time or stop time** of the task.

- **Allocated time quota** for the task.

**Manipulation of task-network resourcing**

It is sometimes required to reallocate human resources in order to manage an evolution incident. Thus, it is necessary to support a change of:

- **Human role** responsible for conducting the task.

---

**CASE STUDY**

This section presents the background and context of the case study, and the project profiles and evolution status of the involved projects.

**Background and Context**

The studied software organization ***XXX*** is a Norwegian banking software house with 326 employees of which 89 are working with software production (a typical SME). It is located in three different sites, with the biggest development department in Trondheim. The proposed case study is carried out closely with a System Development Division in Trondheim. The organization is ISO-9000 certified in January 1995 and thus has a documented quality system. Reporting and tracking procedures are installed to collect project performance metrics such as consumed effort, remaining time and cost on paper-based forms. Still, XXX suffers from late deliveries due to inaccurate estimates. It has also realized that unanticipated changes during project is a major cause. Moreover, there exists neither an empirical base nor a quantitative foundation to assess the possible effects of different types of changes based on previous experience. Most projects running at XXX are homogeneous and similar with respect to contractual conditions, product characteristic, solution architecture with operating platforms, customer profile. In addition, projects adhere to a defined project model which includes contract negotiation, planning, water-fall life cycle (i.e. analysis, design, implement, testing) and experience packaging. The project staff is often synthesized from...
different divisions, depending on the project's required competence/expertise. As the final product is delivered, a corresponding maintenance environment must also be established to assure continuing on-line operation and service. The studied organization needs to have a more sophisticated insight into the evolution profile. Appropriate method and tool support can then be developed. That is the primary objective of the cooperation with our research group. A simulated project environment will be modeled in EPOS and necessary tool support is provided to manage the actual evolution. Lesson learned from the prototyping scenario will be useful for XXX in improving ability in project management.

Project Profile
This Case study is based on five projects named A to E. Following information are retrieved at project start by filling in a project characterization form. Those measures are thus essentially fetched from the XXX's project plan and start report of the studied project. The project profile measures presented in the table below are only a subset of those included in the project characterization form used to gather this information. We only select those project profile measures which demonstrate obvious differences between the five studied projects.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project type</td>
<td>Develop.</td>
<td>Upgrade</td>
<td>Develop.</td>
<td>Upgrade</td>
<td>Upgrade</td>
</tr>
<tr>
<td>Project member</td>
<td>15</td>
<td>14</td>
<td>Varied</td>
<td>8</td>
<td>Varied</td>
</tr>
<tr>
<td>Effort (hour)</td>
<td>6990</td>
<td>3815</td>
<td>1150</td>
<td>1500</td>
<td>1520</td>
</tr>
<tr>
<td>Duration (mth)</td>
<td>13</td>
<td>9</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Customer</td>
<td>Bank</td>
<td>Bank</td>
<td>Bank</td>
<td>Bank</td>
<td>Internal</td>
</tr>
<tr>
<td>Competence</td>
<td>Partial</td>
<td>Internal</td>
<td>Partial</td>
<td>Internal</td>
<td>Internal</td>
</tr>
<tr>
<td>Technology</td>
<td>Partial</td>
<td>Known</td>
<td>Partial</td>
<td>Known</td>
<td>Known</td>
</tr>
</tbody>
</table>

Table 1: Project Profiles for five studied projects

We see from the table that most common project types and sizes at XXX are fairly represented in the project sample. In addition, the degree of risk level which is represented by competence availability, knowledge of applied technology and project difficulty, also cover the entire range of value domains. Projects C and E do not have a fixed number of participants, and personnel are constantly changing during the project life time. That is why their number of project members is defined as varied. Product profile is not included in the table because they are partly not relevant in our further analysis, and partly lacking in the project archive.

Evolution Status
In this section, we present empirical results on evolution which are collected during the case study. A thorough analysis of such data is described in next section. Most information on process evolution are retrieved from the monthly status reports of the five studied projects. Table 2 illustrates the total number of evolution occurrences and their associated correction effort. The average frequency of evolution occurrences per month and average cost for each evolution are then derived.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total no. of evolution</td>
<td>77</td>
<td>19</td>
<td>17</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>No. of evolution per month</td>
<td>2.6</td>
<td>1.2</td>
<td>3.0</td>
<td>2.3</td>
<td>-</td>
</tr>
<tr>
<td>Total correction cost (hour)</td>
<td>1177</td>
<td>504</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Average cost per evol. (hour)</td>
<td>153</td>
<td>27</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Evolution Profiles for five studied projects

Due to lack of evolution data in the project archive, some slots in the table above must be left empty. Complete information on process evolution during project E, and evolution cost in projects C and D are not documented at all. Despite of scarce information, the empirical results above demonstrate an average frequency of evolution occurrences of 2 or 3 per month. An average cost for correction effort is about 90 hours per process evolution. Only projects A and B have fully documented the impact of process evolution. Therefore, we can hardly make further analysis based on such weak data foundation. However, we can demonstrate where process evolution comes from, i.e. where-dimension in the categorization framework. Table 3 presents the percentage of different evolution origins from five studied projects and their average values. The average numbers from the table obviously indicates a superior dominance in percentage of evolution which comes from customer. This conclusion is correct with respect to the perception at XXX. Only now, we have established a quantitative indication to where process evolution comes from.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>34.9</td>
<td>75.0</td>
<td>47</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Sub-contractor</td>
<td>4.8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Executive</td>
<td>72</td>
<td>53</td>
<td>17.7</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Project manager</td>
<td>33.7</td>
<td>15.8</td>
<td>5.9</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>Project members</td>
<td>19.3</td>
<td>5.3</td>
<td>29.5</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3: Percentage of Evolution Origin from five projects

EVOLUTION ANALYSIS
In this section, we describe the results which are obtained by analyzing collected process evolution data. The analysis is basically performed within the context of our categorization framework (as shown in figure 1). A set of typical process evolution patterns are then identified. Empirical relations between process evolution and project profile are also presented. We introduce below two ways to perform analysis by combining arbitrary evolution dimensions in the categorization
framework (i.e. where, why, what, how, when and by-whom).

The following analysis are done by combining dimensions where-why and where-how. The reason for our choice of such combinations is simply that the analysis results demonstrate interesting findings, and are further discussed in subsequent subsections. Of course, there is no restriction on which or on number of evolution dimensions to be combined to perform the analysis.

The numbers which are reported in following tables, are derived from the EPOS Evolution Analyzer Tool (not described in this paper). Only four greatest process evolution origins in table 3 (i.e. Customer, Executive, Project Manager and Project Member) are taken into consideration in the analysis.

Where-Why Frequency Analysis
The percentages in table 4 are derived by keeping one Where-category fixed (e.g. Customer), and then varying the category values in Why-dimension. That is, they illustrate percentages of evolution occurrences distributed over different cause categories. Only interesting categories in the Why-dimension are selected for illustration. The most frequent evolution patterns are identified and discussed in depth later in this paper.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ambiguity</td>
<td>6.5</td>
<td>5.3</td>
<td>17.7</td>
<td>11.1</td>
<td>0</td>
</tr>
<tr>
<td>Error</td>
<td>1.3</td>
<td>5.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Requirement rev.</td>
<td>5.2</td>
<td>15.8</td>
<td>17.7</td>
<td>44.4</td>
<td>25</td>
</tr>
<tr>
<td>Lack of competence</td>
<td>3.9</td>
<td>5.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Delay</td>
<td>15.6</td>
<td>10.5</td>
<td>5.9</td>
<td>11.1</td>
<td>25</td>
</tr>
<tr>
<td>Postponement</td>
<td>5.2</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lack of resource</td>
<td>1.3</td>
<td>0</td>
<td>5.8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Executive</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-prioritizing</td>
<td>7.8</td>
<td>5.3</td>
<td>17.7</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Project manager</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Under-estimate</td>
<td>29.9</td>
<td>15.8</td>
<td>5.9</td>
<td>11.1</td>
<td>25</td>
</tr>
<tr>
<td>Project member</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>11.7</td>
<td>0</td>
<td>17.6</td>
<td>11.1</td>
<td>0</td>
</tr>
<tr>
<td>Lack of competence</td>
<td>5.2</td>
<td>5.3</td>
<td>5.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Delay</td>
<td>3.9</td>
<td>0</td>
<td>0</td>
<td>11.1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Frequency of Evolution Occurrences distributed over Causes

Where-How Frequency Analysis
On the similar manner, the percentages in table 5 are derived by keeping one Where-category fixed (e.g. Customer), and then varying category values in the How-dimension. That is, they represent the percentages of evolution occurrences distributed over different impact categories. Only some interesting categories in the How-dimension are included in the table. The identified impacts are used to validate EPOS' approaches to manage process evolution in section Managing typical evolution patterns in EPOS.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rework</td>
<td>7.8</td>
<td>15.8</td>
<td>0</td>
<td>11.1</td>
<td>25</td>
</tr>
<tr>
<td>Prolong task</td>
<td>7.8</td>
<td>31.8</td>
<td>23.5</td>
<td>33.3</td>
<td>0</td>
</tr>
<tr>
<td>Postpone task</td>
<td>20.8</td>
<td>21</td>
<td>11.8</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Revisit/re-estimate</td>
<td>1.3</td>
<td>5.3</td>
<td>11.8</td>
<td>11.1</td>
<td>0</td>
</tr>
<tr>
<td>Project manager</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rework</td>
<td>11.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Prolong task</td>
<td>7.8</td>
<td>15.8</td>
<td>5.9</td>
<td>11.1</td>
<td>0</td>
</tr>
<tr>
<td>Project member</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rework</td>
<td>7.8</td>
<td>0</td>
<td>11.8</td>
<td>22.2</td>
<td>0</td>
</tr>
<tr>
<td>Prolong task</td>
<td>9.1</td>
<td>0</td>
<td>23.5</td>
<td>11.1</td>
<td>0</td>
</tr>
<tr>
<td>Postpone task</td>
<td>10.4</td>
<td>0</td>
<td>5.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Training</td>
<td>1.3</td>
<td>5.3</td>
<td>5.9</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5: Frequency of Evolution Occurrences distributed over Impacts

The table 5 shows that rework and prolong task are superior evolution impacts on project schedule. Such impacts require additional allocation of resources according to initial plan. The frequency of evolution which requires rework, constitute 23% (summarizing the Rework numbers from table 5), while 36% of total evolution occurrences imply additional resources allocated to a particular task (summarizing the Prolong tasks numbers from table 5). A postponed task due to a process evolution does not involve any additional cost. However, such evolution imply that the plan must be re-scheduled and allocated resources must be released. Still depending on the extent of resource needed to such rescue tasks, the figures in the table clearly demonstrates the importance and seriousness of process evolution.

Typical Evolution Patterns
Based on the figures presented in tables 4 and 5 above, we identify eight most typical and frequent evolution patterns. Such evolution patterns are below listed according to decreasing frequency percentage. Within the description of evolution pattern, we also emphasize the necessary corrective actions. Such rescue actions will be then realized by invoking different operations to manipulate project plan (i.e. task-network) in EPOS (see section Mechanisms for managing process evolution in EPOS).

1. Customer revision (21.6%): The customer issues new or revised requirements according to the initial specification. Such changes requires that completed tasks must be re-executed. The extent of rework depends on which phase in development life-cycle the requirement change is requested. New tasks can also be added into the project plan to deal with new or enhanced system functionalities. Note that the requirement revisions after
initial delivery (i.e. maintenance phase) have not been taken into consideration in this case study.

2. Under-estimation (17.5%): The project manager tends to under-estimate various tasks under planning. This is partly due to misunderstanding of user requirements, and partly lack of empirical foundation for making estimates. Such under-estimation usually causes that initial estimates must be revised, and project plan must be re-scheduled. Sometimes, new tasks are introduced while existing ones are either removed, divided or merged.

3. Customer delay (14%): The customer delays to deliver documents (e.g., requirement, customer site interface) as initially agreed. Such delays require a postponement of tasks which are dependent on the given deliverables. The project plan must then be revised to reflect that fact.

4. Resource re-allocation (11.2%): The executive management (division or senior manager) re-allocates project personnel to other project as a result of strategic re-prioritizing. Such happen when they want to “rescue” other urgent projects, or to satisfy a higher prioritized customer request. This high frequency of staff turnover results in a violation of the project plan. That is, necessary resources and competence are not available to keep in project progress on schedule. Human factors associated to a particular task must be removed, and other dependent tasks must therefore be postponed.

5. Ambiguous requirement (8.1%): The customer is not clear when the initial requirements are specified. This is due partly to their limited insight into the problem, and partly due to their lack of competence in a given technology. That is, important requirement details are neglected or vaguely defined. Such ambiguous requirements usually imply a rework of requirement specification, or an insertion of corrective tasks.

6. Error commitment (8.0%): The project member have injected errors in technical documents. Such an evolution pattern often initiates either rework or prolong the current task. An specialized error removal task can also be inserted to deal with the problem. Such corrective actions imply delay and thus re-scheduling of the project plan.

7. Customer postponement (5.2%): Unexpected requests for postponing the start date of an task are issued by the customer. Such postponement causes that the initial project plan must be revised. The time and human factors which are associated to scheduled tasks are changed.

8. Lack of competence (5.0%): This phenomena is more common among project member (3.2%) than among customer (1.4%). Such events are resolved by assigning project personnel to training to improve e.g. their knowledge on a particular development method/language. Training tasks are thus added to the existing project plan.

Based on the identified evolution patterns above, we have suggested a set of improvement initiatives which are described in the section Managing typical evolution patterns in EPOS.

Empirical Relations between Evolution and Project Profile
In this section, we present our findings on empirical relations between collected data (in previous section) and project characteristics (see table 1). In particular, we observe typical evolution patterns on one hand, and external project profiles on the other hand. Such empirical relations improve our ability to predict and then anticipate process evolution based on what we know at project start (i.e. project profile). We have revealed following interesting relations. Of course, this list is not exhaustive.

Project type vs. Evolution frequency
Development projects (A and C) experience more turbulence than upgrading ones (projects B, D, and E). This fact is illustrated by the high percentages of evolution frequency in table 2 in projects A and C. This phenomena can be explained by the fact that both customer and XXX in development projects deal with a new problem which they often do not have sufficient technical competence.

Project type vs. Customer revision A high degree of customer revisions is found in upgrading projects (B, D, E) in table 4. This observation is somehow hard to explain. There are two assumptions usually associated to upgrading projects. Firstly, technology is well-known, and required competence is available. Secondly, the customer often states clear and well-defined enhancement requirements as they have been using the product for a period of time. However, the second assumption does not hold in our case. For instance in project B, the customer at project start delivered an requirement specification which has thereafter been revised so many times that the initially agreed-upon contract must be completely re-negotiated. Therefore, this empirical relation must be validated carefully later as with more data points.

Project duration vs. Under-estimation The degree of under-estimation seems to increase proportionally with the project duration (see table 4). It is easy to accept the fact that it is hard to predict the behavior of some tasks taking place far beyond in the future. Moreover, project managers at XXX suffer from the lack of empirical baselines from prior projects to rely their estimates on.
The extent of customer delay is proportional to the project duration (see Table 4). This is probably due to the fact that customer tends to give such projects lower priority, and thus does not respect the initially agreed-upon deadlines.

Customers in development projects are often uncertain on what features they really need. This normally leads to unclear and vague specification of initial user requirements.

Table 4 shows a superior representation of errors done by project members in development projects. This is perfectly natural when the project members must address a problem to which they do not completely master.

Development project E with an internal customer (i.e. XXX's Product Division in this case study) has experienced a higher degree of requirement revisions than other projects. This relation is not quantitatively illustrated by the collected data material, but has been acquired through reading other reports. Nevertheless, it is easy to understand when we accept the fact that internal development projects do not bear any economical risk for XXX, and in addition are often given lower priority by the executive management level.

It is worthwhile to stress that the empirical relations above are retrieved by our limited data ground which are collected during the case study. The explanations for the relations are thus somehow speculative. As the amount of collected data grows, more valid relations can then be extracted. After all, our primary intention is to demonstrate the usefulness and benefits from collecting and analyzing such information.

To deal with the typical evolution patterns necessary actions must be made to correct a particular evolution pattern. Most of them imply a revision of the project plan (i.e. task-network in EPOS) by either adding new tasks, manipulating existing ones, changing schedule properties of tasks.

Below we summarize basic operations that can be used to manipulate the EPOS task-network:

**Adding task** Put an additional task to the existing task network.

**Deleting task** Remove an existing task from the task network.

**Schedule revision** Start time, Elapsed time and duration of the task in the network can be revised during execution.

**Resource revision** Performer of the task in the network can be replaced. As the consequence, the task is either suspended or canceled.

Table 6 shows how eight evolution patterns (enumerated 1–8 according to the list presented in last section) are dealt by EPOS' approaches to manage task-network evolution.

<table>
<thead>
<tr>
<th>Evolution pattern</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete task operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Add task operation</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task schedule revision</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Task resource revision</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 6: Evolution Patterns vs. Approaches in EPOS

In this section we state our suggestions for improvements towards the studied software company XXX on the basis of case study results and findings.

**IMPROVEMENT SUGGESTIONS**

**Suggested Improvement Initiatives for XXX**

- The most frequent evolution pattern is revealed to be user requirement revision. XXX should have therefore better dialog with customer in requirement specification phase. That is, advanced technology in requirement engineering can be adopted to better understand requirements and thus prevent them from modifying later.

- The second most frequent evolution pattern is concerned the issue of under-estimation. This problem can be remedied adopting better estimation method. The case study has demonstrated the benefit of learning from past experiences to establish baselines and then improve estimate accuracy. The existing quantitative baselines at XXX are still insufficient and incapable to provide precise estimates. However, they will be improved over time if XXX commits to pursue the same work direction. On the other hand, XXX should initiate better cooperation between System Development Division and Sale Division in contract bidding process. That is, realistic but also competitive bids must be elaborated together.

- Incorporate new paper-based forms for characterizing project/product; collecting evolution data; collecting actual project/product status and categorizing errors into appropriate handbooks in the Quality System. Specifically, the three first forms are included in the Project...
Management Handbook, while the last one in the System Development Handbook. By doing so, XXX institutionalizes an effective data collection process for the company. Further support for data collection and utilization of collected data, can be provided by introducing an Experience Database (EDB). Since an the Experience Database used needs times to be validated, those paper-based forms can be used in the meantime.

- Select one or two pilot projects which will use the EDB for recording project/product measures and process evolution. In parallel, the collected data from the five studied projects should be stored in EDB available for use. The reason to run EDB together with pilot projects is to validate its applicability, user-friendliness and correctness. Necessary enhancements should be recorded and installed before EDB is released for use within the company.

- Project managers are encouraged to use the EDB both to make project estimates, and to anticipate possible impacts during project execution (planning with contingency). It is a major step towards continuous learning and reusing past experiences. As many projects record their data into the EDB, the existing baselines gradually become valid, representative and convergent. The improvement evidences on estimate accuracy are then significantly demonstrated. Patience and full commitment are key issues to establish a valid experience foundation. We refer to an example of at NASA-SEL. It took them 18 years (1977-94) with several experiments to gather data before they manage to consolidate to a set of representative and reliable baselines.

- To be able to achieve significant improvement in error reduction during development, XXX is suggested to seriously collect and classify errors according to their types, origins, severities, and related characteristics (partly covered by the an error report form). XXX can then obtain a deep insight into the error problem and prioritize corrective actions according to error impacts. Adequate technology can thus be evaluated and introduced to reduce number of errors or to detect them earlier. That is, required effort to remove them can be declined considerably.

CONCLUSION

In this paper we have presented a taxonomy for classifying process evolution along six dimensions. A case study has been conducted with a software organization to collect actual evolution data based on the defined categorization framework. The preliminary analysis shows that most changes originate from customer delays or requirement changes. Such changes are detected in the design and testing phase, and constitute a major impact in term of extra cost. A QIP-inspired method with an experience database and associated tools have been developed in EPOS to manage and analyze collected evolution occurrences observed changes. Such support will improve the ability for planning and scheduling to reduce unanticipated changes during project. The gained experiences is fed back to the organization for making effective improvement decisions. However, the reuse of previous project experiences has been negligible since such evolution data are non-existing. More projects will be tracked to make the experience database more confident and complete.

Further, much research effort has dealt with managing changes on process models in term of evolving types or classes. From our new work, we realize that rather pragmatic changes to task instances (instantiated process) are much more frequent than type evolution. The consistency impacts of such instance-level changes need to be better managed and deserve more research attention.

REFERENCES


Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.

ICSE 97 Boston MA USA
Copyright 1997 ACM 0-89791-914-9/97/05 ...$3.50

399