Extended Abstract:

Reuse, Adaptation and Validation of System Development Processes

Peter J. Funk
Mälardalen University, Department of Computer Engineering, Västerås, peter.funk@mdh.se

Abstract. Large companies often use a standard system development process tailored for their special needs and that are locally adapted to different development projects. The advantage is that such a standardised project process ensures measurable progress and quality to meet internal requirements as well as external requirements. The local adaptation to projects needs to address aspects of: project size in time and people, distribution of project staff on different locations, combination/integration of hardware/software and incorporation of local experience, standards and regulations. This local adaptation is a highly manual process requiring skill and time and for large companies starting hundreds of new projects every year it is a considerably large part of the yearly development costs. Unfortunately all the gained experience in such local adaptations are difficult to reuse. Today updating the system development process is made manually by staff responsible for the local adaptation and integrate these experience are integrated into the standard process. The recourses for such work are often not sufficient and the result is often local adaptation that have there own life and result in an out of date standard process where gained experience and local improvements do not spread within the organisation.

Recently a number of formalised graphical notations to describe enterprise models have emerged. These formalisations of the processes enables analysis and comparison of processes. We propose a Case-Based Reasoning (CBR) approach that enables reuse of processes or parts of processes. The CBR method is proposed as an "add on" to existing process development tools such as the MED@X tool (developed at EDT/UAB) and as addition to Rationals RUP (Rationals Unified Processes) when local adaptations and template processes are needed. By proposing a formal graphical notation in which the user can sketch the local development process the CBR matching process can identify and propose reuse of similar processes that are previous local adaptations. Once a new local adaptation has been completed and successfully used it is automatically added to the case library enabling future reuse. The formalised processes also enable validation of the processes. For example, if the requirements are that certain information (documents) should be produced during the project, an automatic analysis of the overall system development process and the individual work flows can determine if these parts are still included after the local adaptation. In the same manner a check that parts from the standard system development process that are not optional are included in the tailoring process can be made. In this paper we analyse the options and investigate related research, outline the MEDAX-CBR approach and propose a three step implementation sketch.

Background

Large companies often use a standard development process used as framework for all their projects. Such a process reflects the companies experience, their commitment to quality and lead time and reflects the category of projects and the level of skill of the members in projects. In small companies an explicit system development processes is rarely used and a projects success and
failure is mostly based on the project leaders and members individual skill and experience. Such skill and experience is expensive to gain if it is in terms of failed projects. More than half of all software projects fail []. Failure often comes as a surprise for management and this after large investments. To reduce this costs large companies invest in a standardised system and product development process used throughout the company, enabling the identification of quality and lead time problems at an early stage by providing clear checkpoints for project members, project leaders and management. As in the case for Ericsson, having tens of thousands of people working in different projects, the savings are believed to be considerable.

**Problem definition**

Ericssons MEDAX process development tool contains Ericssons standard development process (currently seven processes meeting different needs) and enables individual projects to tailor this process to their local needs. All the necessary information from the standardised process are available, with checkpoints, work flow descriptions, information and documents descriptions, templates and examples. Once the standardised process has been adapted to local needs the experience of these adaptations is mostly lost and when it is necessary to update the standard process, tedious interviews and investigations of successful local adaptations need to be made. This results in a divergence between the developed standard model and local improvements. This results in local projects preferring to reuse some previous adaptation of the standard process and base the new project on an adaptation of this process rather than using the Ericsson template process as starting point. By doing so, improvements in the standard process are not transferred to and from new projects via template processes and thereby reducing the value of committing to standard template processes further.

In this paper we address the problem of reusing locally adapted processes by proposing a formalisation of the processes by using a graphical notation based on enterprise modelling standards and work flow standards. Once a local adaptation has been made the process is automatically stored in the global process library. By using standard notations different tools can be used. If a new project is started a description of the type of project is given together with special needs and requirements. These features will be used to identify the most similar locally adapted processes in the process template library which the user can investigate. For the investigation the formalisation enables comparison between different processes and the user is informed of the differences. This is useful to ensure that the proposed and adapted process still meets overall requirements set by internal and external quality and productivity requirements. This approach is embedded in a case-based reasoning approach as described in the next section.

**Case-Based Reasoning**

The central concept of case-based reasoning is expressed by Riesbeck and Schank as: “... the essence of how human reasoning works. People reason from experience. They use their own
experience if they have a relevant one, or they make use of the experience of others ...” [Riesbeck, Schank, 1989, page 7]. Aamodt and Plaza’s picture, Figure 1, illustrates the main ideas of case-based reasoning: a problem is given in the top left corner, similar cases are retrieved from a case library and the most suitable case is selected and re-used. The most suitable case may need to be revised to solve the problem. If the solution is approved, the problem and its solution are stored in the case library. Next time a similar problem is encountered, less adaptation of the retrieved case may be needed and the performance will increase if similar problems are often encountered and the features identifying similar cases are good enough.

![Figure 1: General architecture of a case-based reasoning system. Adapted from [Aamodt, Plaza 94].](image)

Looking at a previous case that has solved a similar problem may, for some situations, be easier to understand because cases provide a context for understanding [Kolodner 93]. A case-based system may also adapt to changing demands, for example, if a new type of problem not previously encountered is solved (if no similar cases are available, a solution to the problem is most likely to be produced manually). The solved problem and its solution are stored in the case library as a new case, with the aim of expanding its competence [Aamodt 93]. The next time the system encounters the same or a similar problem, the system will have increased its potential to produce a solution. It is more likely that, in a rule based system, the rules would need to be updated to include this new class of problems.

Case-based reasoning may be suitable for problem areas in which the knowledge of how a solution is created is poorly understood [Watson 97], e.g. the creation and adaptation of system development process. In technical domains, case-based reasoning has been applied to a variety of application domains such as: architectural design support [Pearce, Goel, Kolodner, Zimring,
Sentosa, Billington, 92]; qualitative reasoning in engineering design [Sycara, Navinchandra, 89], [Nakatani, Tsukiyama, Fukuda, 92], software specification re-use [Maiden, Sutcliffe, 90], software re-use [Fouqué, Matwine, 93], re-use of mechanical designs [Mostow, Barley, Weinrich, 89], [Bardasz, Zeid, 92], telecommunications network management [Brandau, Lemmon, Lafond, 91], fault correction in help desk applications [Watson 97], building regulations [Yang, Robertson, Lee], fault diagnosis and repair of software [Hunt 97]. There are already a few CBR systems in commercial use.

**Implementation Sketch**

In conclusion, case-based reasoning may be applied to application domains that are not sufficiently well understood to create a consistent and complete knowledge-base in how to solve the problem automatically, on condition that:

1. problems and their solutions have similarities.
2. a case library with past problems and their solutions is available or can be created.
3. there are good ways for identifying relevant cases in the case library.
4. solutions can be adapted and re-used for similar problems.

We suggest that development processes fit these requirements well if a formal notation is used. The user does not need to know that there is a formal notation involved and draws the process descriptions in a graphical editor as usual, the editor translates the diagrams to the formal textual representation used in matching and analysis. A matching algorithm that identifies similar behaviour is used [Funk 98]. Traditional comparison and analysis is used for the notations.

The full implementation of this approach is proposed to be carried out in three steps:

1. Implement formalised process descriptions and formalised work flow descriptions. This enables collection of a case library and production of process templates. By using a standard notation (e.g. Process Interchangeable Format, PIF), different tools can be used.
2. Implement some validation tools helping users to improve and secure requirements on the tailored processes. This gives users direct benefits from using the graphical tools.
3. Collect local tailored processes and add them to a case library. Implement comparison, matching and abstraction of processes. This enables transfer and distribution of knowledge from and to template processes.