Process Patterns for Agile Capability Engineering Methodology: The PACEM Project.

Christophe Nécaille
Defence R&D Canada - Valcartier
2459 Pie-XI North, Quebec (QC) G3J 1X5 Canada
christophe.necaille@drdc-rddc.gc.ca

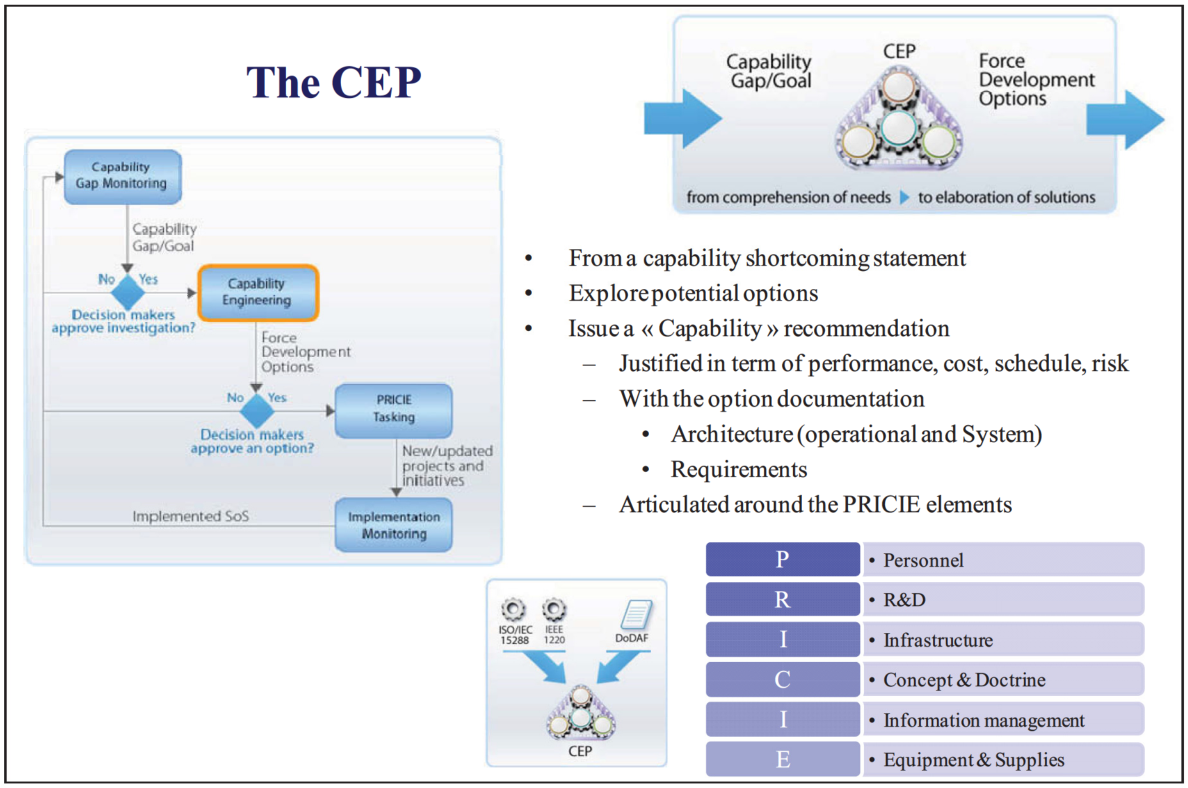
Abstract. This paper presents the approach used to investigate the potential of process patterns to increase the level of agility of engineering processes applied to military capabilities or system of systems. This effort is a follow-up of the Canadian Capability Engineering Process (CEP) defined during the Collaborative, Capability, Definition, Engineering and Management (CapDEM) technology demonstration project. The challenges surrounding process assessment are investigated.

Introduction

Capability Engineering has been the answer of Defence Research and Development Canada (DRDC) to assist the Department of National Defence (DND) in the establishment of Capability Based Planning (CBP) (Pagotto and Walker 2004). More precisely, a Capability Engineering Process (CEP) has been defined in order to reduce the gap between the CBP and the acquisition projects (Bernier et al. 2005). This process develops a set of preferred capability options with a view towards future force structure in answer to an identified gap or affluence. As this process gains in popularity, the diversity of possible applications highlighted the needs for process agility. Following the Collaborative Capability Definition, Engineering and Management Technology Demonstration Project (CapDEM TDP), the PACEM (Patterns and Agility for Capability Engineering Methodology) research project aims to assess the value of process patterns as a mean to improve the agility of such processes. In this paper, the first section starts by setting the background of the project with a short reminder of the CapDEM and CEP results. The presentation of the main concepts used in the project (process patterns and agility) is followed by the project's vision of the integrated engineering methodology platform called "ALADIN"; which is our answer to the need for agility improvement. The strategy and challenges of process improvement assessment are briefly discussed, before opening on perspectives and future work.

Context

The Capability Engineering Process (Lizotte, Lalancette, and Nécaille 2006) has been developed to assist the decision-making process on strategic defense investment or divestment. This process is based on sound engineering standards (ISO/IEC 12288 and ISO 21207) and combines the rigor of systems engineering with the flexibility of iterative and incremental design of its work products. Its primary deliverable is an assessed recommendation of the best force development options to address a capability gap. A force development option is a two-part deliverable, addressing the operational and the system architecture. The system (of systems) architecture part is described according to the facets of the organization (Personnel, R&D, Infrastructure, Concept & doctrine, Information management and Equipment & supplies) designated under the “PRICIE” acronym in Canada and documented according to the Department of Defense Architecture Framework (DODAF) framework (Figure 1)


The CEP
Capability Gap/Goal
CEP
Force Development Options
from comprehension of needs ▸ to elaboration of solutions
Capability Gap Monitoring
Capability Gap/Goal
No
Yes
Capability Engineering
Decision makers approve investigation?
Force Development Options
No
Yes
PRICIE Tasking
Decision makers approve an option?
New/updated projects and initiatives
Implemented SoS
Implementation Monitoring
ISO/IEC 15288
IEEE 1220
DoDAF
CEP

- From a capability shortcoming statement
- Explore potential options
- Issue a « Capability » recommendation
  - Justified in term of performance, cost, schedule, risk
  - With the option documentation
    - Architecture (operational and System)
    - Requirements
  - Articulated around the PRICIE elements

P	• Personnel
R	• R&D
I	• Infrastructure
C	• Concept & Doctrine
I	• Information management
E	• Equipment & Supplies

Figure 1, the Capability Engineering Process

The approach used for the conception of this process (Lizotte et al. 2005), as well as the principles driving its conception (Mokhtari et al. 2005), the process details (Lizotte, Lalancette, and Nécaille 2006), validation strategy (Lalancette et al. 2006), assessment (Robbins et al. 2007) and lessons learned (Lalancette et al. 2008) have been presented in previous INCOSE symposia.

The domain of application of the CEP approach appears to be endless, as it was designed to engineer all kinds of capabilities. Effectively, The CEP application scope went over the domain of strategic capability planning. In 2007 and 2008 several research projects within DRDC showed their interest to use the process to describe and engineer their systems of systems (SoS). The first project used the process during its inception stage, to establish the as-is situation of the capability. A second one had to adapt it in order to deliver the as-is architecture description as soon as possible. The use of CEP by R&D projects confirmed its versatility.

Since CEP validation activities, we were aware of the fact that engineering projects have to evolve in a dynamic context and environment. Funds, availability of stakeholders, classification of information, and even customers requirements are constantly evolving during the lifetime of the project and interact with its execution. A completely defined process from the start of the project is not a perfect answer, and is a pitfall for the adoption of the CEP as it reinforces the perception of a rigid process. On the opposite, allowing free changes of the process workflow during the project increases the risk of losing the rigor it provides. The increasing demand for adapted processes started the reflection on possible means to provide agility to the CEP. Borrowed from the Software Engineering domain, the concept of process patterns appealed to us as it could provide a way to get alternative workflows for the process, and improve its agility. Having all these elements in mind, The "Patterns and Agility for Capability Engineering Methodology (PACEM)" applied-research project was set-up, to assess whether the concept of process patterns can be of any use to improve the agility of the CEP. The first task was to research the literature, then to propose a prototype of an integrated methodology design workshop in which we would be able to document and validate our workflow patterns. Then gathered and collected some process patterns to fill in our repository. The last and most important step will be to validate on assess the potential process improvement, in term of agility.

Agility in SoS engineering methodologies

A CEP Principle. One of the secondary goals of the CEP is to: "Facilitate strategic agility in capability-based planning" (Mokhtari et al. 2005). To reach this goal, a subset of principles called "adaptation principles" (figure 2) has been defined, within the foundational grounds of the CEP.

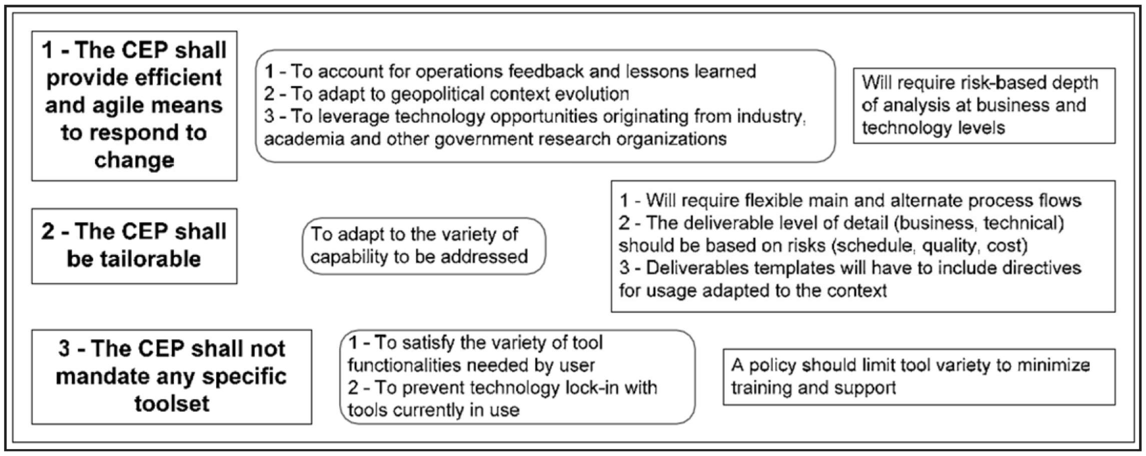
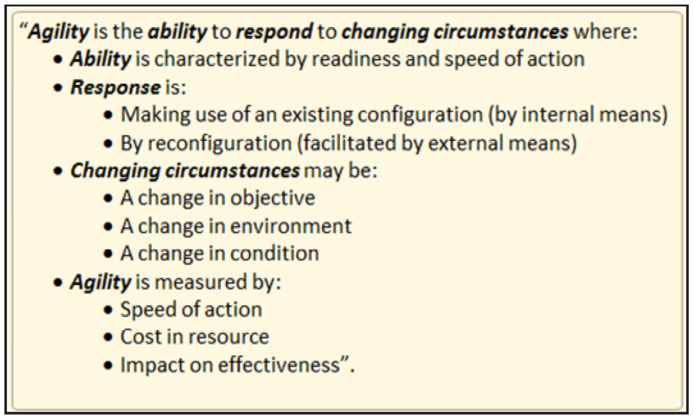

1 - The CEP shall provide efficient and agile means to respond to change
1 - To account for operations feedback and lessons learned
2 - To adapt to geopolitical context evolution
3 - To leverage technology opportunities originating from industry, academia and other government research organizations
Will require risk-based depth of analysis at business and technology levels
2 - The CEP shall be tailorable
To adapt to the variety of capability to be addressed
1 - Will require flexible main and alternate process flows
2 - The deliverable level of detail (business, technical) should be based on risks (schedule, quality, cost)
3 - Deliverables templates will have to include directives for usage adapted to the context
3 - The CEP shall not mandate any specific toolset
1 - To satisfy the variety of tool functionalities needed by user
2 - To prevent technology lock-in with tools currently in use
A policy should limit tool variety to minimize training and support

Figure 2. CEP adaptation principles

Two types of characteristics are mentioned here. The first one deals with the ability to address a variety of capabilities. The process needs to be adaptable prior to its execution. These characteristics define the "static" dimension of agility. The other set of characteristics is related to the "dynamic" dimension of agility or "resilience". In other words, it is the ability of the process to adapt to a variety of external events during its execution, with the minimal loss of performance. Resilience and adaptability are two aspects to be included in the search for agility.

Different definitions. During the initial exploration stage of the project, we realized that the term "agility" has several slightly different meanings depending in which context it is used. In fact, there is no consensus on the meaning and scope of what « agile » means. To add to the confusion, agility has become a buzzword in some domains such has manufacturing and software engineering. After having reviewed several definitions, we particularly appreciated the following one, coined by (Mackley, Barker, and John 2008). – See figure 4 – This definition is applied to network enabled capabilities and has the merit of being well structured.

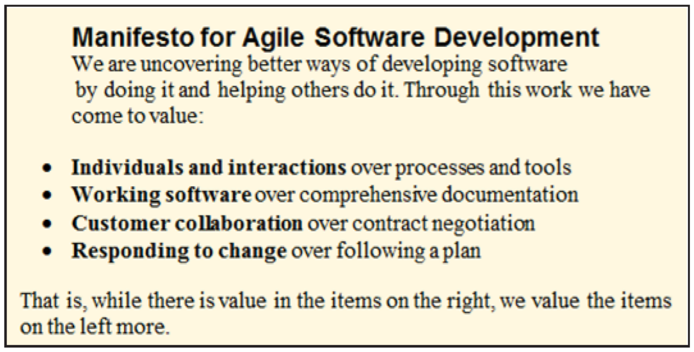


"***Agility*** is the ***ability*** to ***respond*** to ***changing circumstances*** where:

- ***Ability*** is characterized by readiness and speed of action
- ***Response*** is:
  - Making use of an existing configuration (by internal means)
  - By reconfiguration (facilitated by external means)
- ***Changing circumstances*** may be:
  - A change in objective
  - A change in environment
  - A change in condition
- ***Agility*** is measured by:
  - Speed of action
  - Cost in resource
  - Impact on effectiveness".

Figure 4 – NECTISE Agility Definition.

Agile Methodologies. In the field of software engineering, different methodologies can be found, which follow the principle of the "Agile Manifesto". The "Agile Manifesto" (Figure 5) was written in February 2001 by seventeen software development methodologists (who founded the Agile Alliance) in reaction to "heavyweight" traditional engineering methods (Beck et al. 2001).



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 5 Agile Manifesto

This is recognized as the ground-principles of any agile software development methodology. From this manifesto, different methodologies were proposed, such as eXtreme Programming, Adaptative Software Development, Crystal, Scrum, and OpenUP among other, which can be characterized as being iterative, incremental, time-bound and emergent.

Combining Traditional and Agile Approach. The agile characteristics of these methodologies are appealing, but the challenge is to get the benefits of the flexibility and leanness while keeping the rigor of planned engineering approaches. The answer might reside in combining both of the approaches, to "balance" agility and discipline (Barry W. Boehm and Turner 2003). A good illustration of this approach is the Incremental Commitment Model (ICM) (B. Boehm and J. A. Lane 2008). This is a risk-driven framework for tailoring system life-cycle processes which integrates the disciplines of systems acquisition, system engineering and software engineering. This approach uses the notion of risk to determine how much process agility or rigor is enough to satisfy the system's objectives subject to its constraints (Jo Ann Lane and Dahmann 2008). An Agile CEP should follow this shallow path. Since its inception principles, both engineering rigor and agility have been set as foundational goals. Particularly, we intend to increase the adaptability and resilience of the process while taking advantage of the rigor provided by the systems engineering framework.

Patterns and process patterns for SoS engineering

A nice way to introduce the concept of pattern comes from the analogy used by Buschmann et al, in "Systems of Patterns" between patterns and the way experts work (Buschmann et al. 1996). When facing a new problem, experts almost never solve the problem by applying a brand new solution. Most of the time, they recall problems they have solved in the past which may present some similarities with the problem at stake, identify the similarities and the solution they have applied and find ways to adapt part of these experience-validated solutions. Therefore a pattern can be seen as a tripartite relationship between a problem, a context, and a solution description as illustrated in Figure 6. The pattern collects lessons learned, know-how from success-stories and previous experiences.

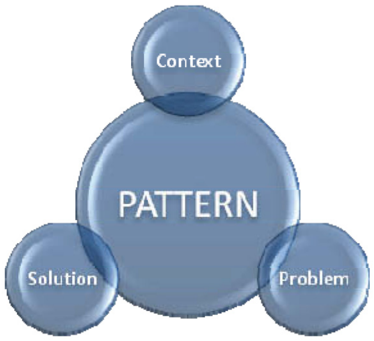


Figure 6 A Triplet: Context, Problem, and Solution.

The strength of patterns resides in their abilities to overpass communication barriers due to different mindset and cultural differences when communicating technical or abstract concepts. They are generic enough to be relevant to many specific problems and the adequate balance and consideration of the forces in presence permits to delineate a specific solution to the particular situation. Patterns have been very popular in software engineering domain. The emergence of object oriented approaches combined with standardized user interface and problem solving reasoning contributed to the discovery of common reusable set of data and functions which were captured in patterns forms (Gamma 1995). The documentation of a pattern usually includes the following parts: name, context and problem description, forces in presence, generic solution, and hints for adapting the solution to particular context. An intriguing characteristic of pattern is that they cannot be invented. Patterns can only be discovered and documented, as they are by nature related to experience. Moreover, a pattern is not really a pattern until it had been experienced by several experts. Therefore, writing good patterns is not an easy task (Appleton 2000) as mastering the generality and relevance of the patterns as well as being able to holistically consider the relevant factors and forces is an art.

Patterns in systems (of systems) and capability engineering. The domain of system of systems engineering has also been hit by the pattern-fever : Different authors have examined the application of process patterns for systems engineering : Barter (Barter 1998), Haskins (Haskins 2005) and more recently Cloutier and Verma (Cloutier and Verma 2007) made different publications defending the thesis that the concept of pattern is applicable to systems engineering. In previous INCOSE symposia, some workshops of pattern writing and discovery had been held (Haskins 2008), giving the participants an opportunity to exercise themselves to documenting patterns. We believe that patterns for capability engineering can depict solutions for capability engineering challenges, in the same way as design patterns works for software engineering. The pattern's solution part can include models and other artifacts of operational and system architectures including requirements and system processes. These architectures parts will depict the inter-relationships of systems and system components both from an operational and systems perspective, and should be documented in such a way that they could be re-used to solve similar capability engineering problems.

Process Patterns. Process patterns, as described by Ambler (Ambler 1998) and Coplien (Coplien and Harrison 2004), are a collection of general techniques, actions, and/or tasks (activities) from which an organization can develop a tailored process that meets its exact needs. Even though Ambler and Coplien originally described process patterns within the context of software development, this concept can be applied to capability engineering. In these patterns, the solution part of the pattern is not a technical design, but a kind of process allowing reaching the solution. For instance, when applying the CEP to a research and development project in a laboratory (the pattern's context) in which management procedure are more informal; the management part of the CEP is way too heavy for research teams (pattern's generic problem). Forces in presence here are in one side the need for a minimal set management procedures (to keep the project on tracks), and on the other side, the desire for leanness and freedom of action of the project's participants. The generic solution is to tailor the process by streamlining management tasks to keep within the process workflow (solution part of the pattern). This pattern is represented in figure 7.

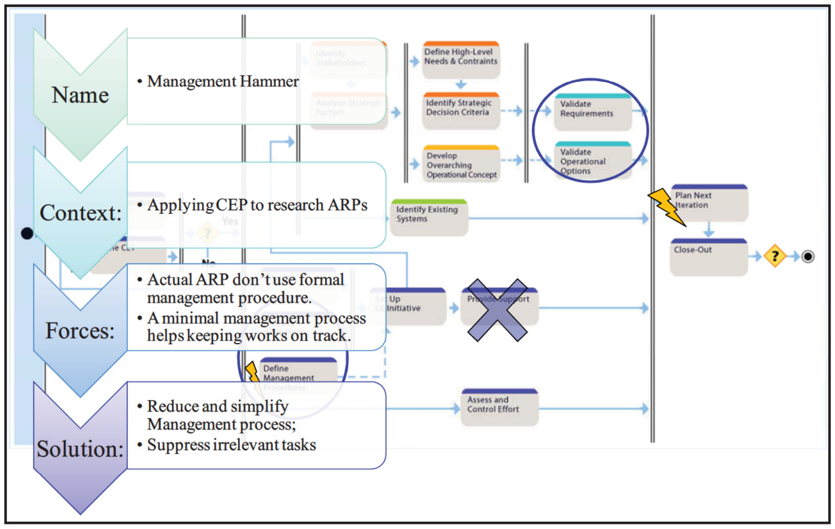

Name
• Management Hammer
Context:
• Applying CEP to research ARPs
Forces:
• Actual ARP don't use formal management procedure.
• A minimal management process helps keeping works on track.
Solution:
• Reduce and simplify Management process;
• Suppress irrelevant tasks
Define High-Level Needs & Contraints
Identify Strategic Decision Criteria
Develop Overarching Operational Concept
Validate Requirements
Validate Operational Options
Identify Existing Systems
Plan Next Iteration
Close-Out
?
Up Initiative
Provide Support
Define Management
Assess and Control Effort

Figure7 Management Hammer Pattern.

From patterns documentation to pattern language. To make an analogy with language, patterns can be seen as the vocabulary of expertise on a domain, while the pattern language is the grammar articulating the different patterns. Considering the execution of a pattern as a transition from a problematic context to a "less problematic" one (in the way that the problem addressed by the pattern should be solved, without considering eventual side effects), a pattern language, in its simplest form, becomes a graph of atomic patterns. Travelling through this graph, one may be able to step from context to context, each step addressing a particular problem. Pattern languages have been proposed for software design (Gamma 1995) and also for process patterns (Coplien and Schmidt 1995).

Patterns for Agile Processes. Whatever methodology is employed, whatever tools or system engineering process is used; projects' successes are before all a people's success (Cockburn 2002). The human factor has a key importance within the patterns. Haskins in (Haskins 2008) shows how patterns provide a good way to consider the human factors within system engineering. This reinforces our base hypothesis behind this work which is that patterns and process patterns are a promising answer for providing agility within a capability or system of systems engineering process. We intend to use patterns to dynamically customize and adapt the process workflow to the environmental conditions of the engineering project, similarly to the approach described in (Hagen and Gruhn 2004). The idea is to identify alternative path of patterns within the process, and to adjust the process accordingly. This will accompany the project team within the maze of the SOS engineering workflow.

We ought to be able to step from contextual state to contextual state, from the beginning of the engineering process until its final task by suggesting the use of patterns relevant to each context. By discovering and implementing process patterns the CEP will be able to adjust itself to the environment, providing what is thought to be the optimized way of reaching the goal.

This section provided a quick overview of patterns, process patterns, and their potential use within systems engineering. The next section will develop our approach and detail the pattern process documentation and implementation within the PACEM project.

Putting all together: Aladin

The vision. While investigating on the potential of process pattern for increasing the level of agility of our capability engineering process, we defined a high-level view of what could be an integrated engineering process engineering environment, named Aladin. Figure 8 details this view, which can be read as a use case.

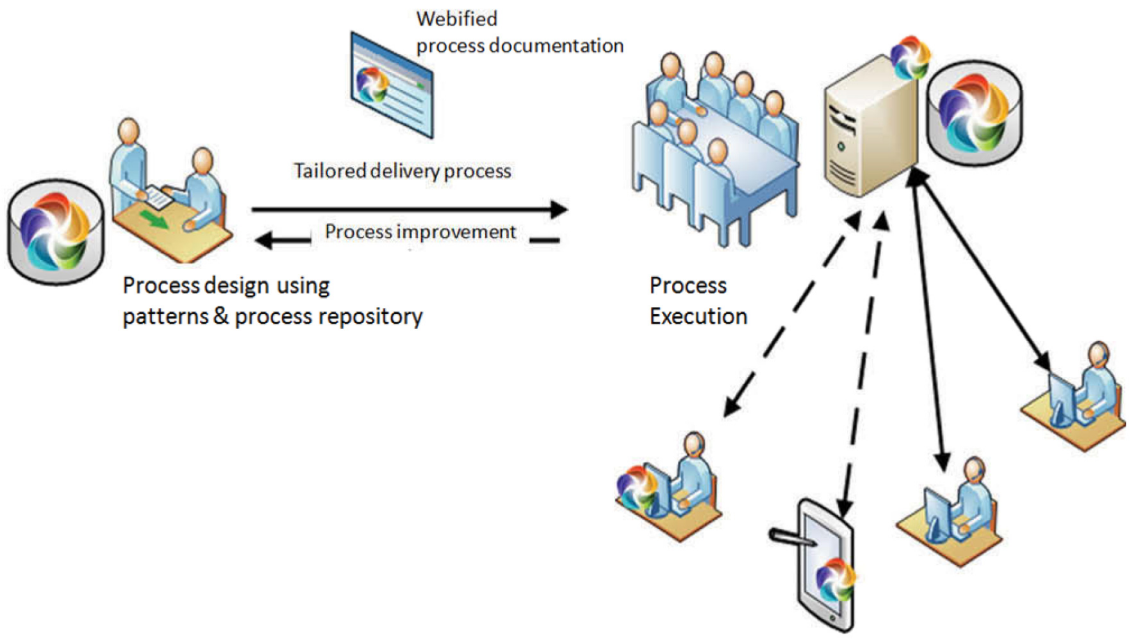


Figure 8: The Aladin overview

From the left, a project team-leader consults the process engineer to adapt the engineering process according to its project's particular context. Selecting from a library of base processes and specific patterns, the process engineer applies the patterns to establish a tailored delivery process. Patterns will provide "static agility" to the process. This process is then deployed to the project team. The team can eventually directly document its work within a process enactment and management tool, whether online or offline. While the team has some flexibility to adapt its process internally, external events may occur and change significantly the process environment, which will result in the need of process adjustment. The CEP is designed to be iterative and incremental, and each of its iterations is surrounded by "gates" providing the team opportunity to ask for feedback and approval on the work in progress. These gates are also a good opportunity for process revision if needed. The team-leader and the process engineer will then reconsider the originally designed workflow. By selecting applicable process patterns, the process evolves in order to react to the event (such has sudden budget cuts, unavailability of team members or stakeholders, etc.). Here, patterns will provide "dynamic agility" to the process.

The platform. To concretize this vision, three categories of tools are needed: tools for process documentation and pattern repository; tools for process enactment and execution management, and tools for process assessment. A quick survey showed that several process enactment tools are available either commercially (IBM Concerto (IBM 2003) from the Rational Jazz suite of tools for instance) or as open source solutions (Bonita, YAWL, etc) (Stoilova and Stoilov 2006). Tools for process assessment will be discussed in a following section; however, our first concern is to be able to document patterns and process workflow within a central repository. Originally, the CEP was manually published by the mean of a website, which contained the hyperlinked description of workflows of tasks, artifacts and deliverables produced as well as the description of the roles performing these tasks (Lizotte et al. 2009). With a need to get all process documentation within a single repository and a concern for standardization, we made a survey of the different standards available for process documentation and used an open source product, Eclipse Process (EPF) (Haumer 2007), for storing the process documentation. EPF is an open source platform for process engineering which basically offers two functionalities: first, It is a tool for documenting and storing a knowledge base of methodological content (method content, which is basically description of tasks, roles, artifacts, techniques ...). This knowledge base can be then retrieved, adapted, augmented with guidelines, definition etc. Secondly, EPF permits to assemble a delivery process by selecting and tailoring parts of the knowledge base. This process can be published on a website or even a wiki platform for collaborative design. EPF is based on the Eclipse platform. The knowledge base is structured according to the Unified Method Authoring schema, which is compliant with the Software process Engineering Model (SPEM) and the Business Process Modeling Notation (BPMN). UMA categorizes the knowledge and separates the static from the dynamic point of view by separating the method content from the process workflow, as shown in the following figure, from (Haumer 2007):

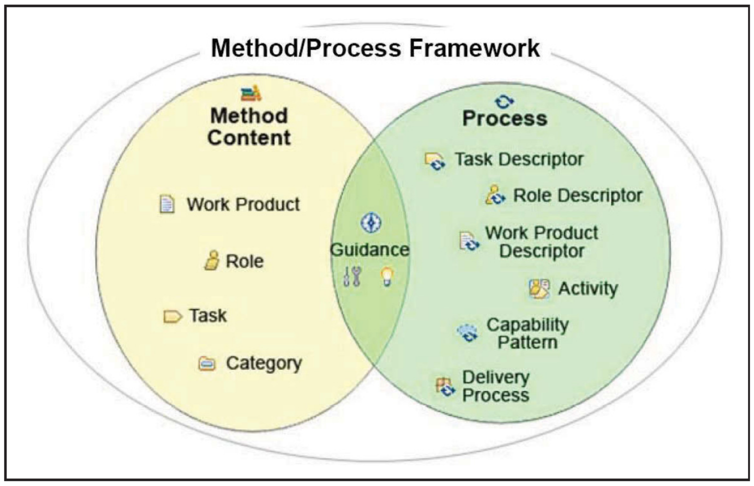


Figure 9 Separating method content from process behavior.

Process Pattern Implementation. The UMA specification already contains a concept called capability pattern similar to the one we wanted to implement. A capability pattern is a reusable chunk of process workflow. However, the capability pattern documentation within EPF is software oriented, and lacks to capture some important aspects of project context. These missing attributes will permit to categorize the patterns according to different criteria applicable to the CEP context. They will be used to detect whether the pattern would be applicable or not in a specific contrext. We developed our own template, and implemented it within EPF, as a separate "Process Pattern" class derived from the "Capability Pattern" class. Some searching and filtering features were added to EPF, to be able to search and quickly find and select patterns relevant to a given context.

Process pattern documentation. The template used for pattern documentation is derived from the numerous pattern templates available such as in (Cloutier 2006). As in almost every pattern description template, it documents each facet of the pattern (problem, context and solution) as well as the forces in presence. Some particular elements related to the capability engineering context have been added within the context description. The amount of experience of the designers in applying the process, the level of availability of stakeholders or the part of the process schedule in which the pattern may apply are a few example of them.

Process patterns for capability engineering processes

A first round of process pattern gathering has been done, firstly by going back to the lessons learned from previous CEP application (Lalancette et al. 2008) and also by interviewing experimented practitioners. This effort led to the description of holistic process patterns, i.e. patterns that affect all disciplines and apply to the whole process. The following paragraphs provide a quick overview of some of them.

CEP high-spin. This pattern is a rapid turnaround capability gap recommendation method useful when the capability gap is fluid. Effort is dispensed in short bursts. The decision to proceed with the recommendation or to proceed through a second, or nth, sprint can then be made on the basis of the recommendation and additional information that might have come up during the sprint. The pattern adds velocity to nominal CEP. It minimizes effort spent on investigating improbable capability options.

This pattern applies in the context where a knowledgeable client representative is available on site, and has the authority to take immediate decisions concerning the options. Up to date information is available and subject matter experts (SME's) are available full time. The Capability engineering team (CET) is small, is knowledgeable of the capability domain and has several previous experience of applying the CEP.

There are no iterations, as defined in the nominal CEP, or even stages per se. Rather, the artefacts are completed informally and all at the same time in each sprint or time box. Options definition, requirements definition, operational and SoS architectures are developed in rapid sprints. The option is iteratively refined through a second, third or additional sprints until the client agrees to the option recommended.

CEP Given Option. A "given option" situation happens when the client has already decided on a solution to a capability gap. If the option is given, there is normally no need for a CEP initiative. However, the client may be interested in learning more about the architecture (operational and SoS) and also about the cost and effectiveness of this one given option. The nominal CEP could be used but without the options selection tasks. This would constitute a holistic pattern because the major disciplines are affected. The relationship from operational options to SoS options is a "many to many" relationship. Hence, there are three possibilities:

- One given operational option with one given SoS option. This is the most restrictive situation;
- "One given operational option" that may lead to several "SoS options"; or
- "One given SoS option" for which there could be several operational options".

These three different cases are documented as three related patterns. The proposed solution simplifies the nominal CEP workflow by removing the activities and tasks related to option selection.

CEP-Lite family of patterns. Each of the CEP discipline (management, requirements engineering, operational architecture, SoS architecture …) has a "lite" related pattern, to be used when the resources are sparse in the corresponding field. The pattern keeps only the barebones activities and work products, enabling the project to be adjusted according to the limited availability of some roles within the team.

As-is First. The nominal CEP builds and evaluates the "As-is" option in parallel with the "To-be" options, in order to keep the efforts spent on the "As-is" to the minimum amount required. However, in several projects the customer requested to get the "As-is" documentation and assessment as quickly as possible, even if the "As-is" option was going to be rejected within the process execution. This required a few changes within the nominal process to accommodate this particular request.

Early close-out. The last pattern to be briefly described here is the "early close-out" pattern. This pattern applies in the context of an abrupt request to terminate the project in the middle of its execution. The aim of this pattern is to provide a generic workflow to close-out all project activities while trying to backup the work that has been done, for eventual future reuse.

Most of these patterns are "high-level" patterns, replacing completely the original workflow. While they answer particular project requirements, they might not be well suited to be combined together. We are investigating now patterns with a smaller scope, for easier combination of patterns.

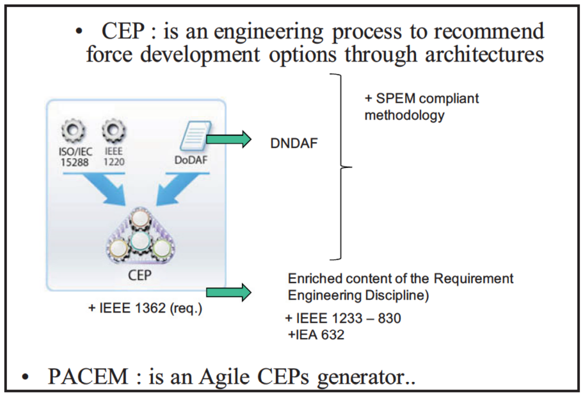
Assessing the assembly

To validate the value of patterns, we first made an exploration of existing process performance parameters and measures, before establishing an assessment strategy. The aim is to be able to determine whether the use of process pattern will enhance or increase the overall process agility. Primary performance indicators were chosen (efficiency, velocity, effectiveness and of course agility) and defined, and a metric chain has been defined. Different methods were considered (CMMI, Performance Dashboard, INCOSE Measurement Primer (Antony et al. 1998)) and were amalgamated together to provide a process assessment framework using earned value reports.

Another thread of work has been started, to explore the option to use simulation for theoretical validation of the patterns. Validation of process patterns through experimentation as described in (Estabraghy and Dalcher 2007) or (Germain and Robillard 2008) certainly provides the most accurate results, but has a prohibitive cost in term of time and resources (Amescua et al. 2006). Moreover, experimentation has its own defects, as it is almost impossible to avoid biases caused previous experience and cultural background of the team members. How to provide two identical contexts to two instances of the same project? (Wohlin, Runeson, and Höst 2000). Therefore, theoretical validation is needed to simulate process execution and analyze possible deadlocks or unused resources, prior to considering a pattern as a potential candidate. Approaches using colored Petri networks (Jensen 1997) for collaborative process execution assessment such has described in (Cai et al. 2005) are under review, to complement the toolset we need for process improvement assessment.

Conclusion

Process patterns are a promising way to allow an engineering process to adapt itself according to external events, while keeping it under control. In this paper, we have presented our view on using process patterns for this usage. We have implemented the pattern support and a first round of pattern discovery and documentation has been done, which permit to evolve from a single capability engineering process toward an agile engineering process generator (Figure 11).



- CEP : is an engineering process to recommend force development options through architectures

ISO/IEC 15288
IEEE 1220
DoDAF
CEP
+ IEEE 1362 (req.)
DNDAF
+ SPEM compliant methodology
Enriched content of the Requirement Engineering Discipline)
+ IEEE 1233 – 830
+IEA 632

- PACEM : is an Agile CEPs generator..

Figure11 : CEP Evolution

The next step is to perform a process assessment activity to have a first idea on the real advantage brought by process patterns. Several methods for process agility improvement assessment have been presented, and the final selection of the method has not been done yet. Future improvement of our platform may now focus on the "execution" part of the processes: While the actual process adjustment through the application of process patterns has to be made manually, it is possible to envision a dynamic adjustment of the process. A process execution monitoring environment would be able to trigger patterns and dynamically adjust the process according to process execution indicators. The process then dynamically adapts itself to its evolving context, which open the way for self adapting, auto-evolving patterns, and why not, automatic pattern discovery?

References

1. Ambler, S. W. 1998. Process Patterns. Cambridge University Press.
2. Amescua, A., J. Garcia, M. Sanchez-Segura, and F. Medina-Dominguez. 2006. A pattern-based solution to bridge the gap between theory and practice in using process models. Lecture Notes in Computer Science 3966: 97.
3. Antony, P., J. Dunn, W. Farr, D. H. Rhodes, G. J. Roedler, C. Tilton, and E. R. Widmann. 1998. Systems Engineering Measurement Primer. INCOSE Measurement Working Group.
4. Appleton, B. 2000. Patterns and Software: Essential Concepts and Terminology. `http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html`.
5. Barter, R. 1998. a systems engineering pattern language. In Proceedings of the 8th Annual International Symposium of the International Council on Systems Engineering (Vancouver, B.C.). Seattle: INCOSE.
6. Beck, K., M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, et al. 2001. Manifesto for Agile Software Development. `http://agilemanifesto.org/`.
7. Bernier, F., M. Mokhtari, M. Couture, M. Lizotte, F. Lemieux, S. Lam, C. Lalancette, and G. Dussault. 2005. The Axes Guiding Development of a Capability

Engineering Process. In Proceedings of the 15th International Annual Symposium of the International Council on System Engineering (Rochester, NY). Seattle: INCOSE.

8. Boehm, B., and J. A Lane. 2008. Guide for Using the Incremental Commitment Model (ICM) for Systems Engineering of DoD Projects, Version 0.5 (USC-CSSE TR-2009-500). Center for Systems and Software Engineering, University of Southern California.

9. Boehm, B. W., and R. Turner. 2003. Balancing agility and discipline: a guide for the perplexed. Addison-Wesley, August 21.

10. Buschmann, F., R. Meunier, H. Rohner, P. Sommerlad, and M. Stal. 1996. Pattern-oriented software Architecture : Vol 1 A system of Patterns.:Wiley

11. Cai, J., S. C-Y. Lu, F. Grobler, M. Case, and N. Jing. 2005. Modeling and managing collaborative processes over the internet. Business Process Management Journal 11, no. 3: 255 - 274..

12. Cloutier, R. J. 2006. Applicability of patterns to architecting complex systems. Doctoral dissertation, Stevens Institute of Technology.

13. Cloutier, R. J., and D. Verma. 2007. Applying the concept of patterns to systems architecture. Systems Engineering 10, no. 2: 138-154.

14. Cockburn, A. 2002. Methodologies as swimsuits. `http://alistair.accountsupport.com/index.php/Methodology_as_Swimsuits`

15. Coplien, J. O., and N. B. Harrison. 2004. Organizational Patterns of Agile Software Development. Prentice Hall PTR.

16. Coplien, J. O., and D. C. Schmidt. 1995. Pattern Languages of Program Design. Reading Mass.: Addison-Wesley

17. Estabraghy, A., and D. Dalcher. 2007. A Controlled Experiment to Investigate the Effect of `Process Patterns' on the Quality of Requirement Analysis. In Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications, 2007., 645-649.

18. Gamma, E. 1995. Design patterns : elements of reusable object-oriented software. Reading Mass.: Addison-Wesley.

19. Germain, E., and P. N. Robillard. 2008. Towards software process patterns: An empirical analysis of the behavior of student teams. Information and Software Technology, Vol 50, no. 11 (October): 1088-1097.

20. Hagen, M., and V. Gruhn. 2004. Process patterns-a means to describe processes in a flexible way. ProSim04, Edinburgh, United Kingdom.

21. Haskins, C. 2005. Application of Patterns and Pattern Languages to Systems Engineering. In Proceedings of the 15th International Council on System Engineering International Annual Symposium (Rochester, NY). Seattle: INCOSE.

22. ———. 2008. Using patterns to transition systems engineering from a technological to social context. Systems Engineering 11, no. 2: 147-155.

23. Haumer, P. 2007. Eclipse Process Framework Composer Part 1 Key Concepts. `http://www.eclipse.org/epf/general/EPFComposerOverviewP`

art1.pdf.

24. IBM. 2003. IBM Rational Team Concert 2.0.0.2 - Help. May 16. http://publib.boulder.ibm.com/infocenter/rtc/v2r0m0/topic/com.ibm.team.concert.doc/helpindex_rtc.htm.

25. Jensen, K. 1997. A brief introduction to coloured Petri nets. in proceedings of Tools and Algorithms for the Construction and Analysis of Systems: 203–208: Springer.

26. Lalancette, C., M. Lizotte, C. Nécaille, W. Robbins, and B. Waruszynski. 2006. Toward the Institutionalization of Capability Engineering. In Proceedings of the 16th Annual International Symposium of the International Council on System Engineering (Orlando, FL). Seattle: INCOSE.

27. Lalancette, C., M. Lizotte, C. Nécaille, W. Robbins, B. Waruszynski, and J. Pagotto. 2008. Capability Engineering within Canadian Defence: Experimentation and Lessons Learned. In Proceedings of the 18th Annual International Symposium of the International Council on System Engineering (Utrecht, NL). Seattle: INCOSE.

28. Lane, J. A., and J. S. Dahmann. 2008. Process evolution to support system of systems engineering. In Proceedings of the 2nd international workshop on Ultra-large-scale software-intensive systems, 11-14. Leipzig, Germany: ACM.

29. Liu, L., D. Russell, N. Looker, D. Webster, J. Xu, J. K. Davies, and K. Irvin. 2008. Evolutionary Service-Oriented Architecture for Network Enabled Capability. In Second International Workshop on Verification and Evaluation of Computer and Communication Systems VECoS 2008 (Leeds, UK). London: British Computer Society.

30. Lizotte, M., C. Lalancette, G. Dussault, S. Lam, M. Couture, M. Mokhtari, and F. Bernier. 2005. A Meta-process Producing a Deliverable-centric Process. In Proceedings of the 15th International Annual Symposium of the International Council on System Engineering (Rochester, NY). Seattle: INCOSE.

31. Lizotte, M., C. Lalancette, and C. Nécaille. 2006. Capability engineering for strategic decision making. In Proceedings of the 16th Annual International Symposium of the International Council on System Engineering (Orlando, FL). Seattle: INCOSE.

32. Lizotte, M., C. Necaille, C. Lalancette, and G. Dussault. 2009. Capability Engineering Process Version 2008 (CEP 2008). Technical Report TR2008-266. Valcartier: Defence R&D Canada - Valcartier.

33. Mackley, T., S. Barker, and P. John. 2008. Concepts of Agility in Network Enabled Capability. In RNEC Conference 2008. Leeds, UK.

34. Miller, G. 2001. The characteristics of agile software processes. In Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS39). Washington DC: IEEE Computer Society.

35. Mokhtari, M., M. Lizotte, S. Lam, C. Lalancette, G. Dussault, M. Couture, and F. Bernier. 2005. Canadian Capability Engineering Process Foundation. In Proceedings of the 15th International Annual Symposium of the International Council on System Engineering (Rochester, NY). Seattle: INCOSE.

36. Pagotto, J., and R. S. Walker. 2004. Capability Engineering - Transforming Defence Acquisition in Canada. In Proceedings of SPIE Defense & Security Symposium, 5441:89-100. Orlando, FL, USA.

37. Robbins, W., C. Lalancette, M. Lizotte, C. Nécaille, J. Pagotto, and B. Waruszynski. 2007. Capability Engineering: Learning from Practice. In Proceedings of the 17th Annual International Symposium of the International Council on System Engineering (San Diego, CA). Seattle: INCOSE, June 24.

38. Stoilova, K., and T. Stoilov. 2006. Comparison of workflow software products. In CompSysTech. Proceedings of the International Conference on Computer Systems and Technologies.

39. Wohlin, C., P. Runeson, and M. Höst. 2000. Experimentation in software engineering: an introduction. Springer Netherlands.

Biography

Dr. Christophe Nécaille is a Defence Scientist within the Systems of Systems section of Defence R&D Canada in Valcartier (Québec) since 2004. His current research interests include process engineering, as well as systems of systems engineering methodologies. He participated in the Capability Engineering Process development and leads the "PACEM" project which aims to improve process agility through the application of process patterns. He holds a Ph.D. in Computer Science from Université de Rouen (France). His previous academic background includes a Msc. in Artificial intelligence and a Bsc. in Electrical engineering.