

Maturity is also about the Capability to Conform the Process to the Right Context!

Mira Kajko-Mattsson
School of Information and Communication Technology
Royal Institute of Technology
Sweden
+46-8-16 16 70
mekm2@kth.se

ABSTRACT

Organizations, their businesses and contexts are multi-dimensional, diverse, and very complex today. Hence, creating homogenous process models for managing them may not always be an optimal solution. Instead, organizations should be able to tailor their processes to the formality level required for the context at hand. In this paper, we claim that the organizational maturity is not only about how organizations are capable to manage their processes. It is also about how capable they are in adapting them to specific contexts and business needs. We also suggest *Context-Driven Process Orchestration Method* (CoDPOM), based on the concept of *practice choreography* and *process orchestration*. The CoDPOM's role is to aid software practitioners in identifying process needs and in recognizing waste which, in turn, would aid them in adapting their software processes to specific contexts, business needs and formality levels.

Categories and Subject Descriptors

D.2.9 [Management]: Software Process Models.

General Terms

Management, Documentation.

Keywords

Software process orchestration, practice choreography, silver bullet, adaptation capability, process backbone, context-driven process orchestration method (CoDPOM), waste, leanness, agility.

1. INTRODUCTION

Companies have many processes that are run in parallel and/or in sequence. To manage them, they have defined homogeneous organization-wide generic process models and guidelines to be reused in various project and non-project related contexts. Soon, however, they have realized that the process contexts strongly vary and that there is no such thing as one homogeneous software process model that fits all the heterogeneous contexts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

FoSER 2010, November 7–8, 2010, Santa Fe, New Mexico, USA.

Copyright 2010 ACM 978-1-4503-0427-6/10/11...\$10.00.

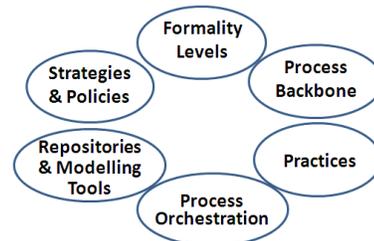


Figure 1. CoDPOM components.

To embrace heterogeneity, software organizations have created many process variants that are dedicated to various needs, development styles, product complexities, process formalities, cultures, and project types [2], [3]. This is, however, not an optimal way of defining processes and maximizing business and productivity results. It creates a challenge of how to manage a broad portfolio of processes, how to choose among them and how to mine and reuse knowledge and experience from them. It also creates a challenge to track the generic processes, identify their variants, and thereby, extract knowledge and experience from them in order to effectively reuse them in the future. All this hampers organizations from improving their processes and makes them continuously reinvent the wheel [9].

Forcing individuals to follow standardized and homogeneous process models within organizations may sometimes have a negative impact on their creativity and productivity. Many times, attempts to make the processes compliant with process models get in the way and slow down the production pace. They also strongly impact the software engineers' motivation for conducting their chores in some specific contexts. [1]

It appears that there is no universal formula for choosing the right processes for the right contexts and formality levels. Organizations, their businesses and contexts are multi-dimensional, diverse and very complex. Using homogenous process models for managing them may not always be an optimal solution. For this reason, we feel that the software community needs a formula for identifying process needs and for recognizing waste which would aid them in adapting their processes to specific contexts, business needs and formality levels. The new formula should also allow the companies to assess the organizational maturity from the perspective of how effective the organizations are in adapting their processes to a particular context and formality level.

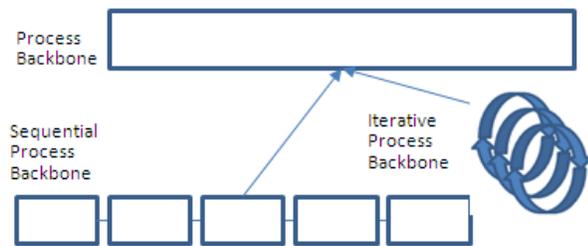


Figure 2. Core process backbones.

In this paper, we claim that the organizational maturity is not only about how organizations are capable to manage their processes, but also how capable they are in adapting them to specific contexts and business needs. The organizational maturity is dependent on how organizations are capable of accumulating and reusing their process knowledge in various process-related contexts in order to maximize their business results and minimize process missteps. Hence, it should also be strongly dependent on how they are capable to orchestrate their processes and practices so that they match the needs and contexts of current business objectives, needs, quality requirements, and market trends.

In this paper, we also suggest context-driven process orchestration method (CoDPOM) based on the concept of *practice choreography* and *process orchestration*. The scope of CoDPOM applies to all types of processes: primary lifecycle processes, supporting lifecycle processes and various management processes [6].

The remainder of this paper is as follows. Section 2 presents CoDPOM method. Section 3 raises questions that need to be researched on. Finally, Section 4, rounds up this paper by claiming that the CoDPOM method might be one of the silver bullet solutions that the software community is striving for today.

2. CoDPOM

It is essential to choose the right set of activities for the right process, its context and formality requirements. Hence, when creating process instances to be executed in a specific context, CoDPOM considers six elements. These are *Process Backbone*, *Practices*, *Process Orchestration*, *Repositories and Modeling Tools*, *Strategies and Policies*, and *Formality Levels* (see Figure 1).

2.1 Process Backbone

Process backbone provides a process template for all the processes. It is the most sustaining process part constituting a common

denominator for all the processes. It provides a foundation for choreographing the practices.

As illustrated in Figure 2, we suggest one generic process backbone and two specialized ones, sequential and iterative. The reason to why we have chosen those two styles is the fact that they are the most commonly used styles in the industry today. They also constitute a platform for defining other more specialized process styles, if need arises.

Processes do not need to exclusively follow a specific sequential or iterative process style. The styles may be mixed in many contexts. An example is provided in Figure 3 where the *Sequential Process Backbone* constitutes a template for defining a business cycle process instance, whereas the *Iterative Process Backbone* constitutes a platform for a highly iterative implementation process instance. The belonging to a specialized. process backbone process is not exclusive. The iterative backbone process may be part of the sequential process and vice versa

2.2 Practice

The next core element in creating process instances is practice. We define practice as a way of working that has been developed through knowledge and experience gained when developing and maintaining software systems. Practices are core elements in creating processes.

As illustrated on the left hand side of Figure 4, each practice is described with eleven properties. However, the contents of *Properties 3-11* are driven by the first two properties which are *Context* and *Formality Level*. It is their values that determine the following:

- Process using the practice.
- Activities belonging to the practice at hand.
- Information required for managing the practice.
- Measurement covering the measurement goals relevant for the practice and its formality level.
- Documentation needs relevant for the formality level required.
- Experience reporting on knowledge, feedback or skills gained while being involved in or exposed to the practice.
- Expertise and roles required for performing the practice.
- Policies and strategies for creating the practice.
- Guidelines for how to choreograph the practice with other practices.

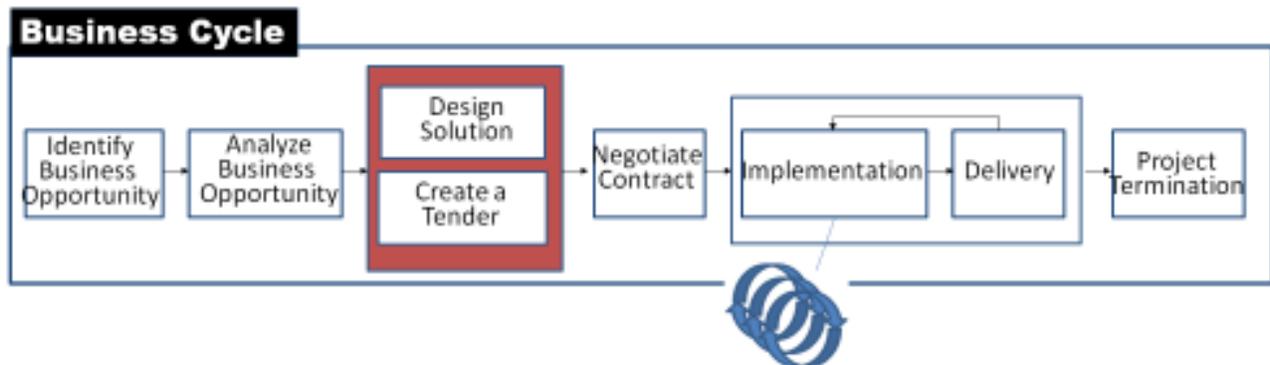


Figure 3. A simplified illustration of mixing process styles [7].

Practice Description	Process Orchestration
<ul style="list-style-type: none"> • Context • Formality level • Process • Activities • Information managed • Measurement • Documentation needs • Experience • Expertise • Policies and strategies • Choreography instructions 	<ul style="list-style-type: none"> • Context • Formality level • Choice of practices • Order of practices • Rules and recommendations • Measurement • Documentation needs • Experience • Expertise • Process meta models • Scalability guidelines • Policies and strategies • Process orchestration guidelines

Figure 4. Practice and Process Orchestration Structures.

2.3 Process Orchestration

The process backbone is the central part of a process. It corresponds to a practice container. It is method neutral. A specific process instance is realized by choreographing existing software practices relevant for the context at hand [5]. A specific process instance is created on an as-needed and context-driven basis. Depending on the context at hand, the orchestration may result in heavyweight, middleweight or lightweight process instances or a mixture of those.

The organizations should feel free to decide when to bind their practices when orchestrating their processes. Both early and late bindings should be allowed. In the contexts dealing with many uncertainties and unknowns, we recommend as late binding as possible. In other contexts, practices may be early bound to specific process instances. However, in cases of unexpected situations, they should be easily unbound and rebound. The reasons may be emergent changes to be made to the systems or processes, changed contexts, changes prerequisites, changed needs, changed understanding, and the like.

To orchestrate processes is not easy bearing in mind the fact that processes may comprise a great number of practices, they may need to follow specific organizational strategies and policies, they may have to consider the context and changes within it and they may have to adhere to specific formality levels. In order to obtain processes that are suitable for specific contexts and needs, the organizations should compose processes by extracting and assembling practices. To maximize process results and to minimize process missteps, waste and failures, organizations need information supporting them in their process orchestration work. Such information is briefly presented on the right hand side of Figure 4. It includes thirteen different properties whose contents are driven by the first two properties which are *Context* and *Formality Level*. It is their values that determine the following:

- List and amount of practices chosen for a specific process.
- Order among the practices chosen for the orchestrated process at hand.
- Rules and recommendations to be followed when orchestrating and performing the process.
- Suggestions for the overall process measurement.
- Documentation needs relevant for the formality level required.
- Experience reporting on knowledge, feedback or skills gained while being involved in or exposed to the orchestrated process variant.

- Expertise and roles required for orchestrating and performing the orchestrated process.
- Meta models describing processes and their variants.
- Policies and strategies for creating the process.
- Orchestration instructions providing guidelines for how to combine practices.
- Scalability guidelines describing how the process can be shrank or expanded to fit the context at hand.

The roles involved in orchestrating and performing processes vary with the context as well. This is because the scope of CoDPOM method covers all the lifecycle processes, including primary lifecycle processes, supporting lifecycle processes and organizational lifecycle processes [6]. Hence, the choice of roles involved does not only depend on the context and the mandate of the roles involved but also on the processes to be orchestrated, their formality requirements and the formality requirements for their inherent practices.

2.4 Supporting Process Orchestration

Software organizations need to be supported with various organizational policies, strategies, repositories and modeling tools. Defining policies and strategies and considering them in process orchestration is crucial for succeeding when choosing the right process instance. Organizations must have policies defining shrewdness and prudence for a specific course of business action and strategies comprising carefully devised plans for acting towards achieving specific business goals.

Information on policies and strategies is very important for choosing the right practices and for orchestrating the appropriate processes. Especially important is it to stress that policies and strategies continuously change and adapt to the changing business environment. For this reason, CoDPOM covers the capability to create, change and enact new and/or existing policies and strategies.

Orchestrating processes is not trivial. First, it requires modeling tools for shaping the processes. Second, it requires relevant information and repositories recording process information. Third, it requires that the repositories be integrated with CASE tools so that they can (1) easily provide feedback while orchestrating and executing processes and (2) record current process information to provide experience and lessons learned for the purpose of future process orchestrations.

2.5 Process Formality and Maturity

The business objectives and contexts at hand do not always require high process formality. In some contexts, the organizations may be content with lower formality levels. For this reason, they need a context-driven adaptation method aiding them in determining process needs and in recognizing process waste.

To achieve context-driven process adaptation, we suggest that all the practices be defined on several formality levels and the processes be orchestrated according to the formality needs required. It is only then the organizations achieve process flexibility by choosing the appropriate process formality for their process instances.

Our concept of practice choreography and process orchestration influences the concept of organizational maturity to produce software. In our opinion, *organizational maturity is not only about how organizations are capable to manage their processes.*

Organizational maturity is also about how organizations are capable to conform their processes to the right contexts.

3. RESEARCH QUESTIONS

Our concept of process orchestration and orchestrated maturity influences the concept of process reuse, flexibility, leanness, scalability, completeness and coherence. Now, the practices may be reused in various contexts and on different formality levels. The process is highly flexible because it is orchestrated for a specific context. The process is highly scalable because the choice and amount of its practices is adapted to the current business, formality needs. The process is lean because it only uses the practices that are right for the context at hand [11]. When orchestrating processes, CoDPOM automatically excludes all waste. However, one should keep in mind that what is waste in one context may be an important prerequisite process element in another context. Our method is complete because it makes certain that the process only covers the practices that are required for the context at hand. And, finally, it is coherent because it creates logically consistent processes that match the current context.

Our method solves the documentation and measurement dilemma. By matching the documentation and measurement needs to a specific process formality requirements, it determines how much we should document and measure. It leads to higher stakeholder satisfaction as well. Individuals may feel confident that they perform realistic processes including the right tasks required for the right context. They may also be aware when to exploit their ingenuity and creativity or when to accept that the process rigidity does not allow enough space for any innovative or imaginative abilities [1].

Our method facilitates contract negotiations. The parties involved may agree upon the formality levels required for developing and maintaining software. This, in turn, may provide input to cost estimations. Finally, our concept suggests a framework delineating what is permitted and what is not permitted when orchestrating a process for a certain context and maturity level.

With our suggestion for a CoDPOM method, we do not claim that current process maturity models are useless or irrelevant. On the contrary, we believe that CoDPOM may complement the maturity models by providing guidelines for how organizations may tailor their process [4], [8], [10]. Also as illustrated in Figure 5, maturity models are an important ingredient in CoDPOM method.

Implementing CoDPOM method may not be easy. Right now, we have many questions that need to be researched on. Some of them are the following:

- When orchestrating processes, how do we find appropriate accuracy and level of detail?
- How do we reuse practices in an effective way?
- How do we meaningfully apply practices in the best way for a given context?
- How do we effectively scale up and down the process?
- How do we define waste and how do we value it in different contexts?
- How and in what contexts should we encourage individual ingenuity, creativity, and teamwork, and reuse of process assets?
- Can practices on different formality levels be choreographed in one and the same process instance?

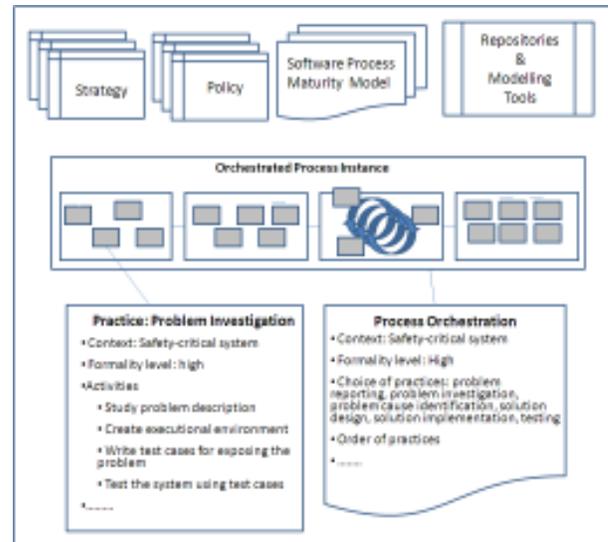


Figure 5. Ingredients in CoDPOM.

- How do we define formality levels for the practices and the processes?
- How should we assess formality of the process consisting of practices on various formality levels?
- How do we evaluate organizational maturity based on the organization's capability to adapt its processes to a specific context?
- Do we need specialized process backbones?
- How do we orchestrate a process so that it fits specific formality needs?
- How do we support process orchestration with process modeling techniques and tools?
- Who owns the process and who owns the practice?
- How should we capture experience and lessons learned of a specific practice and orchestrated process?
- How do we know that the orchestrated process is the right one?
- How do we improve practices and processes?
- How do we monitor and control the process?
- How do we embrace uncertainty?
- And many other questions to come.

4. EPILOGUE

For years, the software community has tried to tackle the problem of rigid and inflexible processes, and for years, the software community has tried to find ways for making them more relaxed and flexible. We regard our CoDPOM method as a silver bullet solution to this problem. It aids the software organizations in supporting their needs for adapting their processes to the specific contexts, needs, and formality levels. In our opinion, it is the understanding of the context and the adaptation capability that is a token of organizational maturity. We strongly believe in our method and we strongly recommend that the software community continue elaborating on it. It is only in this way, we may see whether it is one of the potential silver bullets to be soon fired.

5. ACKNOWLEDGEMENTS

We would like to thank Mr. Paul McMahon, Mrs. Winifred Menezes and Mr. Luigi Buglione for their valuable comments on the definition of maturity.

6. REFERENCES

- [1] Baker, S.W., Formalizing Agility, Part 2: How an Agile Organization Embraced the CMMI, *In Proceedings of AGILE Conference*, 2006, IEEE Computer Society, 147-154.
- [2] Beck, M., Managing Process Diversity while Improving Your Practices, *IEEE Software*, Vol. 18, Issue 3, IEEE Computer Society, 21-27.
- [3] Bollinger, T., McGowan, C., A Critical Look at Software Capability Evaluations: An Update, *IEEE Software*, 26, 5, (Sept.-Oct. 2009), 80-83, DOI=10.1109/MS.2009.119.
- [4] Carnegie Mellon and SEI, Capability Maturity Model Integration (CMMI), <http://www.sei.cmu.edu/tools/index.cfm>, retrieved on June 6, 2010.
- [5] Cockburn, A., Methodology Per Project, <http://alstair.cockburn.us/Methodology+per+project>, retrieved on June 8, 2010.
- [6] ISO/IEC 12207: 2008, Systems and Software Engineering - Software life cycle processes, 2008.
- [7] Kajko-Mattsson M., Sjökvist K., Söderström J., DRiMaP - A Model of Distributed Risk Management Process, in Proceedings of Fifth International Joint Conference on INC, IMS and IDC, ISBN: 978-1-4244-5209-5, IEEE, 2009.
- [8] McMahon, P., *Integrating CMMI® and Agile Development*, Addison-Wesley, ISBN: 978-0-321-71410-7, 2010.
- [9] Nyfjord, J., Kajko-Mattsson, M., Wengelin, D., Exemplary System Development Framework Needed! Position Paper, http://www.semat.org/pub/Main/WorkshopPositions/SEMATA_position_SAAB.pdf, retrieved on June 6, 2010.
- [10] Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B. Capability Maturity Model SM for Software, Version 1.1. *Technical Report* (Carnegie Mellon University / Software Engineering Institute). CMU/SEI-93-TR-024 ESC-TR-93-177, 1993, <http://www.chc-3.com/class/tr24.93.pdf>, retrieved on August 15, 2010.
- [11] Poppendieck, M., Poppendieck, T., *Implementing Lean Software Development: From Concept to Cash*, Addison-Wesley Professional, ISBN:0321437381, 2006.