A BPM Lifecycle Plug-in for Modeling Methods Agility

Wilfrid Utz, Robert Buchmann, Dominik Bork, Dimitris Karagiannis

AMCIS 2020 Proceedings

Organizational Transformation & Information Systems (SIGORSA)

Aug 10th, 12:00 AM

A BPM Lifecycle Plug-in for Modeling Methods Agility

Wilfrid Utz
BOC Asset Management GmbH, wilfrid.utz@univie.ac.at

Robert Buchmann
Babes-Bolyai University, robert.buchmann@econ.ubbcluj.ro

Dominik Bork
University of Vienna, dominik.bork@univie.ac.at

Dimitris Karagiannis
University of Vienna, dk@dke.univie.ac.at

A BPM Lifecycle Plug-in for Modeling Methods Agility

Completed Research Full Papers

Wilfrid Utz
BOC Asset Management GmbH, Austria
wilfrid.utz@boc-eu.com

Robert Andrei Buchmann
University Babeş-Bolyai, Romania
robert.buchmann@econ.ubbcluj.ro

Dominik Bork
University of Vienna, Austria
dominik.bork@univie.ac.at

Dimitris Karagiannis
University of Vienna, Austria
dk@dke.univie.ac.at

Abstract. Business Process Management literature has proposed several BPM lifecycles on a level of abstraction that is "modeling method"-agnostic, i.e. they consider the modeling language and tool support an underlying invariant or technological concern. While remaining on the same abstraction layer, we highlight a "method agility" requirement observed in commercial BPM consulting projects - concretely, it manifests as change requests for the modeling language or tool, from one lifecycle iteration to the next, leading to situations of "model value co-creation" as customer demands are assimilated in the modeling method. Based on a conceptualization of such situations, a lifecycle "plug-in" is proposed in the form of a methodology and associated tool support, allowing for responsive evolution of the adopted modeling method with impact on several lifecycle phases. Historical examples from the evolution of a BPM product are provided to illustrate and classify the demands that motivate the existence of this "lifecycle plug-in".

Keywords

BPM lifecycles, modeling method agility, modeling method requirements.

Introduction

Business Process Management (BPM) emerged as an alternative to functional management, by proposing a process-centric view on organizations, to directly enable improvement of how work is performed. In order to map business-process-managing processes and to ensure a commitment of improvement continuity, a variety of BPM lifecycles have been conceptualized (de Morais et al., 2014).

The goal of this paper is to bring to light, in this context, a *method agility requirement* that informs a reconsideration (and hereby proposed extension) of traditional BPM lifecycles. The work is based on observations from consulting experience with BPM projects and tool deployment for customers around Europe – which will be kept anonymous, while highlighting some of their requirements leading to the conceptualization proposed in this paper: a BPM lifecycle modification is thus proposed in the form of a "plug-in" subprocess that addresses the aforementioned *method agility requirement*. Results of its successful long-term application have been assimilated into an evolving BPM product.

Being language-agnostic, BPM lifecycles consider the business process modeling method to be invariant between consecutive lifecycle iterations – i.e. the process conceptualization is assumed to be fixed and stable, result of a prior decision of adopting one or another notation or modeling toolset. The lifecycle abstraction neglects possible on-the-fly customizations of the modeling method. On the other hand, the scientific literature on Conceptual Modeling has acknowledged a recurring requirement for "agility on modeling method level" (Bork et al. 2019) – an adaptation of agile software development principles to the practice of modeling method engineering and modeling tool customization. This *method agility requirement* manifests when end-users of a modeling tool raise explicit requests for the re-engineering of that tool – something that typically propagates to changes in the underlying modeling language, in order to satisfy case-specific needs. Traditionally, such situations have been relegated to the fields of Domain-specific Modeling Languages or Domain Engineering (Kelly and Tolvanen, 2008; Reinhartz-Berger et al.

2013; Karagiannis et al., 2016), with the assumption that such customization requests typically lead to a specialization of modeling concepts for a narrow domain. Researchers have also proposed the notion of "situational methods" (Henderson-Sellers et al., 2014) – which, instead of emphasizing specificity, refers to the necessity of reconstructing methods from existing "chunks" in order to meet engineering demands for particular situations. Even more specific for this paper's scope, the notion of "method requirements" is presented in (Gupta and Prakash, 2001) as "high-level abstraction of services that a method will provide and constraints under which it functions"; this is further specialized to "modeling method requirements" in (Karagiannis et al., 2019), where authors propose tools for managing this class of requirements that is rarely recognized explicitly in Requirements Engineering literature. All these are examples of recognizing the *method agility requirement*, whose position relative to BPM lifecycles is made explicit by this paper.

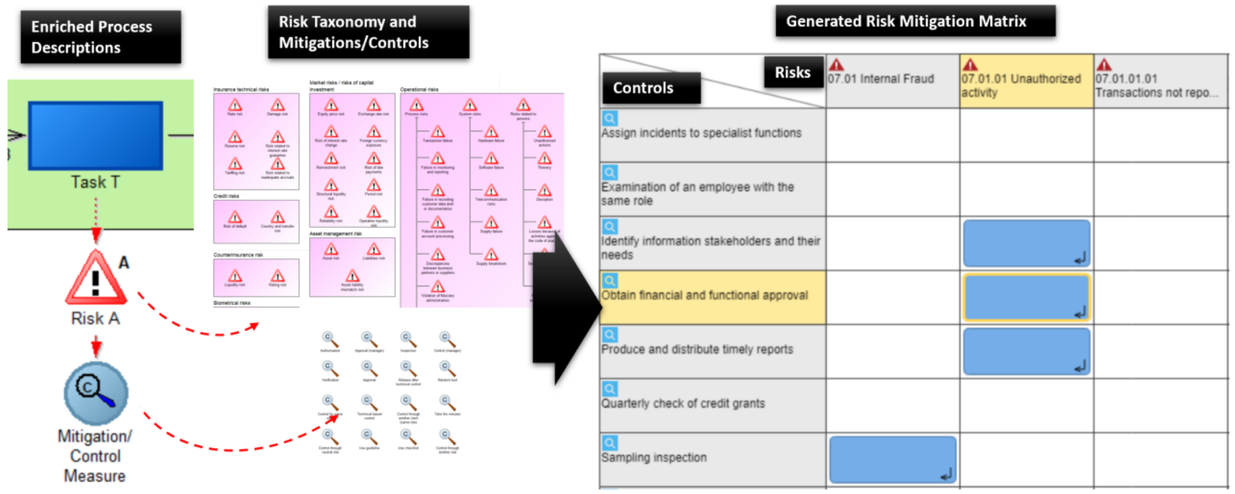
The remainder of the paper is structured as follows: the next section formulates the Problem Statement. It is followed by comments on Related Works. Then the proposed BPM lifecycle enhancement is introduced – firstly through a high-level overview, followed by explanations of the nature of modeling method requirements that motivates the proposal. Further on, the methodological nature of this enhancement is discussed in detail and related to real case examples from BPM consulting projects where it was applied.

Problem Statement and Background

The most prominent BPM lifecycles proposed in the literature, inventoried and compared in recent works (de Morais et al., 2014; Szelagowski, 2018), do not make explicit the modeling method agility requirement, nor a dedicated lifecycle subprocess to address it. However, when confronting such conceptual lifecycles with pragmatic projects, the *method agility requirement* reclaims certain escalation activities for the customization of the process modeling method. This aspect is reflected by the Conceptual Modeling literature in its "domain-specific" segment, but not transferred explicitly to BPM lifecycles, presumably because lifecycles are considered modeling method-agnostic. However, the adequacy of lifecycles is occasionally challenged, when additional detail must accommodate certain disruptors - e.g. the Process Mining Manifesto (van der Aalst, 2011). Such changes aim to improve the streamlining expressed by lifecycle models, their support for BPM continuity; the resulting lifecycle enrichment informs more precise BPM project planning, the development of BPM product-service systems and introduces new patterns to be scrutinized by the scientific literature.

According to (Gartner, 2020), “A business process coordinates the behavior of people, systems, information and *things* to produce business outcomes in support of a business strategy." The term "things" suggests an open-ended recognition of case-specific assets that may enrich process designs during collaborative, consulting-based modeling that can be considered a form of "model value co-creation". The process modeling language and tool must occasionally accommodate customer demands for capturing enterprise-specific details – from superficial aspects (graphical ornamentations) to the introduction of new concepts, new types of models or model-based functionality. Co-creation has long been recognized as a value enhancement strategy (Kambil et al. 1999) and became a core principle of Service-dominant Logic (Vargo and Lush, 2004). This paper advocates a particular interpretation in the context of BPM consulting services, where customers are co-interested in reengineering their process modeling method, thus extending traditional BPM lifecycles with a "plug-in" subprocess that enables enhancements between consecutive lifecycle iterations. Such enhancements are not limited to model annotations, they include (a) *language adaptation* (capturing some enterprise culture specifics), (b) *extension* (adding new concepts and types of models), or (c) *integration* (between different types of models or between models and model-driven functionality).

Even if the practice of business process modeling is well supported by a variety of established languages (EPC, BPMN etc.), method change requests can originate in (a) how a BPM suite or system evolves or in (b) the competency questions that models should be able to answer (via e.g. reporting, model queries, reasoning). Regarding the first point, BPM systems take direct input from process repositories and are driven by the knowledge schema governing those repositories (i.e. the metamodel of the process modeling language). Regarding the second point, competency questions are a typical motivator for knowledge systems development (Wisniewski et al., 2019), and if we see BPM through the lens of that paradigm, as in (Maier, 2004), we have to recognize that the process description language may need to assimilate richer semantics as competency questions evolve. The current paper will point to customer requirements and their transformative impact on the BPM lifecycle; satisfying those change requests is not only a matter of patching tools, but also of reconceptualizing the modeling language - to achieve the desired specificity and to deliver value for evolving modeling purposes. For example, a Risk Management use case may need reporting features that rely on the ability of modelers to attach to process elements Risk events and mitigation Controls, according to some enterprise-specific taxonomy (see Figure 1).


Enriched Process Descriptions
Risk Taxonomy and Mitigations/Controls
Generated Risk Mitigation Matrix
Task T
A
Risk A
Mitigation/ Control Measure
Controls
Risks

Controls \ Risks	07.01 Internal Fraud	07.01.01 Unauthorized activity	07.01.01 Transactions not repo...
Assign incidents to specialist functions			
Examination of an employee with the same role			
Identify information stakeholders and their needs		🟦	
Obtain financial and functional approval		🟦	
Produce and distribute timely reports		🟦	
Quarterly check of credit grants			
Sampling inspection	🟦		

Figure 1. Exemplary modeling customization in response to Risk Management "method requirements"

Related Works

A BPM lifecycle has "the role of a map of the business-process-managing process in the organization" (Szelagowski, 2018) and may undergo changes due to technological specificities or recurring processes observed in practice: the recent trends of Robotic Process Automation propose novel lifecycles that contain software development elements, e.g. (Bizcon, 2019); the Process Mining Manifesto (van der Aalst, 2011) proposed, through the lens of process intelligence use cases, introducing the possibility of "adjustment during execution". This paper's proposal positions agile adjustment on the reengineering/redesign phases, from a metamodeling perspective. Independently of technological disruptions, multiple BPM lifecycles have gained popularity - the paper (Szelagowski, 2018) points to variants discussed in (PNM SOFT, 2017; Pourshahid et al., 2009; Di Ciccio et al., 2015, Bernardo et al., 2017) and even proposes a novel lifecycle to reflect BPM dynamics observed in practice. We hereby specialize the notion of dynamic BPM by considering responsiveness to modeling method requirements, manifested as an escalation path observed in commercial projects during some lifecycle iterations.

Coming from another research direction, Moody's work on notation improvements and shortcomings (Moody et al., 2009) raised awareness that visual notation is subject to optimization and requirements. Its "physics of notation" also falls under the "modeling method requirements" class that motivate this work. The need for modeling method or language customization has been approached from several directions outside BPM, in the Conceptual Modeling literature – e.g. the MetaCase tooling of (Kelly et al., 2013) or the requirements process for domain-specific modeling languages (Frank, 2013). These are, however, not linked to BPM lifecycles and their hereby proposed extension. The knowledge acquisition phase, necessary for domain experts involved in method engineering is hereby subordinated to a management of modeling method requirements for the purpose of model value co-creation. Customer participation and dedicated roles are essential to capture granular demands and to make sure that they are agilely reflected in modeling method increments from one iteration of the BPM lifecycle to the next.

Agile Modeling Method Engineering

BPM Lifecycle Extension Points

BPM lifecycles are relatively stable conceptual artefacts emerging from practice or scientific literature, positioned on an abstraction layer that is agnostic of modeling tool/method and commonly serve as framing for BPM research work. However, variability and evolution of those underlying tools/methods create escalation situations for which lifecycle management must be prepared regardless of how radical the customization must be and on what language it is applied to. This paper does not advocate lowering the abstraction of BPM lifecycles to accommodate particular modeling methods (examples will be given only for illustration purposes) - the goal is to highlight opportunities for BPM tool servitization that may be dismissed as deployment details, but are in fact recurring subprocesses in need of dedicated support. For this purpose, the Agile Modeling Method Engineering metamodeling framework (Karagiannis, 2016) is grafted as a lifecycle "plug-in" on prominent lifecycles from the BPM literature:

- Lifecycle A, proposed in (Karagiannis et al., 1996) – it specifies several processes (Strategic Decision, Reengineering, Resource Allocation, Workflow Management) and a feedback loop; its key characteristic is that a link between BPM and strategy is established and multiple levels of detail unfold from strategy to execution. The Reengineering process is where modeling is primarily performed;
- Lifecycle B, detailed in (Dumas et. al, 2016) – its emphasis falls on process discovery and conformance checks, assuming that a convenient standard method was adopted for process (re)design and analysis; process hierarchies are implied by the Process architecture artefact, which means that different levels of specificity may be necessary, but the lifecycle does not dedicate a step to enable this;
- Lifecycle C, proposed by Gartner (Polancic, 2013) – it is inspired by Quality Control methods, as a specialization of the DMAIC (Define-Measure-Analyze-Improve-Control) cycle, extended with explicit recognition that a modeling method is employed for certain phases.

Figure 2 shows each of these lifecycles extending with an escalation path labelled as *Agile Modeling Method Engineering* (AMME) – which provides the lifecycle "plug-in" whose details will be zoomed in the subsequent sections.

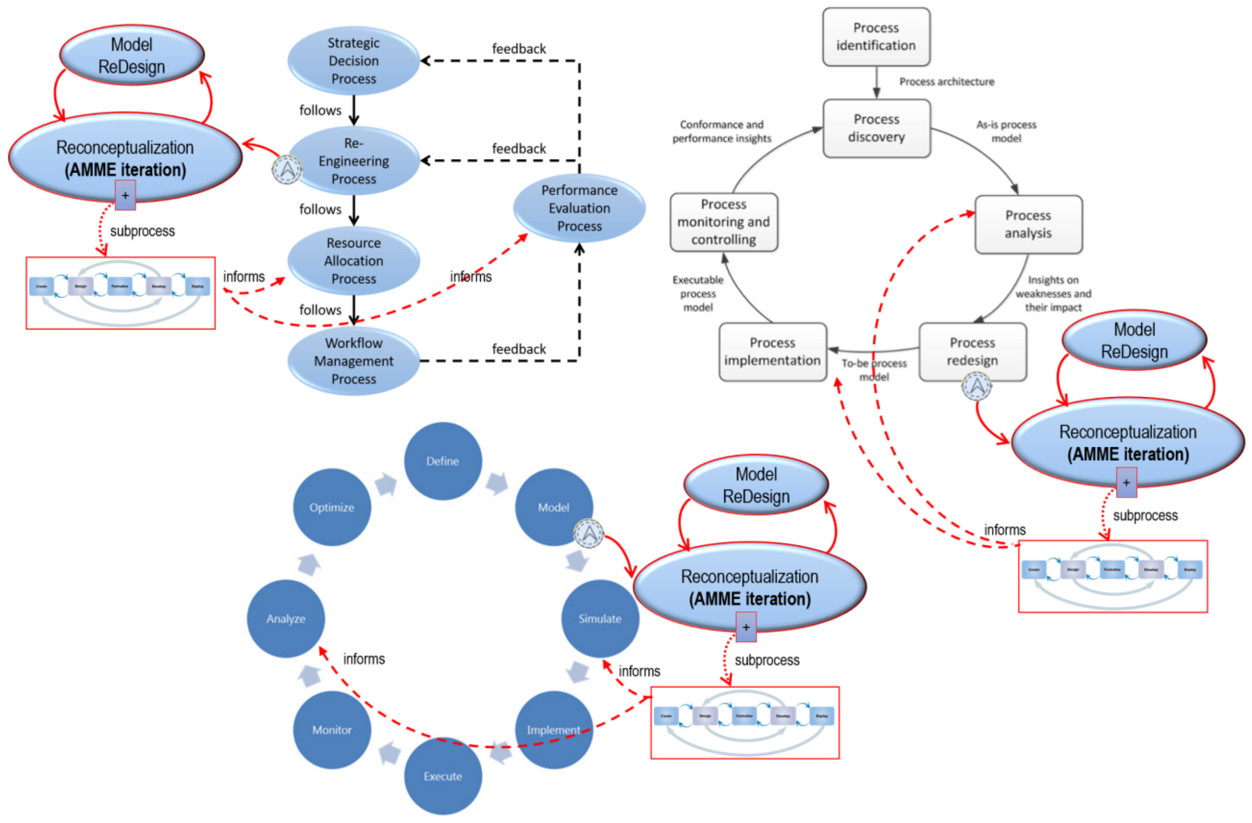


Figure 2. Plugging in AMME iterations to Lifecycle A (top-left), Lifecycle B (right), and Lifecycle C (bottom)

In this section we only highlight the docking points between AMME and the original structure of these lifecycles, as well as to indicate the nature of their interfacing: the AMME escalation manifests either during *modeling phases* (when modelers fail to express some case-specific detail, then fall back to textual annotation or commenting that later become semantic change requests) or during *re-engineering phases* (when process reengineering reclaims not only improvements in the process structure, but also semantic enrichment for more informed decision making or improved process-aware functionality). The AMME reconceptualization results not only in minor tool tweaks, but also in modeling language adaptations that will inform other lifecycle phases, e.g.: (a) the Resource Allocation and Evaluation processes in Lifecycle A – popular business process modeling languages limit their notion of resources to participants, documents and IT systems, while case-specific resources may also need to be depicted; (b) the Process Analysis and Implementation phases in Lifecycle B – these are decision making and deployment phases that may need again to take into consideration case-specific properties and resource types that have been abstracted in the modeling language; (c) the Simulate and Analyze phases in Lifecycle C – for similar reasons.

For the purposes of this paper, we have collected observations from applying the BPM tool ADONIS (BOC, 2020b) in commercial consulting projects. The practical experience, with escalation cases where the tool needed to meet case-specific customer demands, inspired this paper's proposal of revisiting and extending BPM lifecycles in a way that has proven its operationality. Some customer demands lead in time to novel product features; others are limited to case-based customizations not deemed sufficiently reusable for commercial adoption - but of interest for Design Research. The next subsection will delve into the nature of "modeling method requirements", then the AMME "plug-in" will be presented together with exemplary changes that it brought to the ADONIS product (BOC, 2020b).

Making Modeling Method Requirements Explicit

Conceptual Modeling specificity can manifest on different layers – we hereby make a proposal of such layers, which can serve as a frame for classifying modeling method requirements: (a) *Domain-specific* – this is the generic term related to language specialization, well established in the Conceptual Modeling literature (Frank, 2013), usually in dichotomy with General-purpose modeling; (b) *Technology-specific* – this kind of specificity is motivated by a need for interoperability with some narrow technological conditions in the context of model-driven engineering; see an example of a process-modeling tool incorporating semantic technology concepts in (Harkai et al., 2018); (c) *Case-specific* – we consider this the narrowest kind of specificity, where a single enterprise raises granular requirements that are only relevant within its operational context. The Domain-specific/General-purpose dichotomy gets a new treatment as the aforementioned *method agility requirement* may translate to either shifts in purpose or in depth of specificity, which manifest in a continuous Purpose-Domain space as reflected in Figure 3.

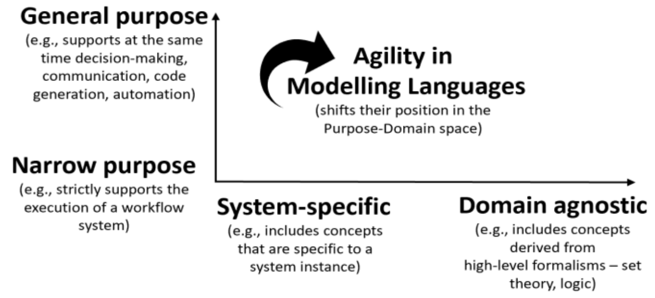

General purpose
(e.g., supports at the same time decision-making, communication, code generation, automation)
Agility in Modelling Languages
(shifts their position in the Purpose-Domain space)
Narrow purpose
(e.g., strictly supports the execution of a workflow system)
System-specific
(e.g., includes concepts that are specific to a system instance)
Domain agnostic
(e.g., includes concepts derived from high-level formalisms – set theory, logic)

Figure 3. The "Method Agility" requirement may shift model specificity (Buchmann et al., 2019)

Business process modeling tools are a subclass of Conceptual Modeling tools and can be subjected to similar shifts, oftentimes from one iteration of the BPM lifecycle to the next. This paper picks some exemplary modeling method requirements from the history of the BPM product ADONIS (BOC, 2020b). One reason for making such requirements explicit and distinct from traditional RE categories is that they raise particular traceability and dependency challenges – e. g. if a stakeholder needs a new feature in a model-driven system, this may translate not only to a functional requirement on the system, but also propagates to new concepts in the modeling language, new graphical symbols and syntactic rules on diagrammatic level, changes in the modeling procedure or in the designer component. Therefore it is important to distinguish such requirements according to the building blocks of a modeling method, for which we take as reference the work of (Karagiannis and Kuhn, 2002): (a) *Modeling language*, comprising notation, syntax, semantics as interdependent building blocks that define dual model value –

both as visual structures for a human audience and as machine-readable knowledge structures to be queried and reasoned upon by machines; (b) *Modeling procedure*, comprising the steps necessary to achieve a modeling goal (e.g. simulation, automation); (c) *Model-based functionality* comprising modeling software features that make use of structure and semantics of model contents. In the following, we point to several common change requests and how they propagate between those building blocks:

- **Model-driven systems need to query machine-readable model contents** (nodes, connectors, editable properties), therefore the modeling language (metamodel) is perceived by such systems as a model schema. Similarly to how database schemas may need to expand in order to satisfy changing requirements and enable richer query results, the schema of a process repository may also need to expand – provided that the change can be rapidly prototyped into the modeling tool (and this is where the AMME "plug-in" comes into play). Model-based functionality also includes mechanisms implemented in the modeling tool itself – e.g. report generation, workload simulation, process path analysis; these also take input from model contents and are constrained by the schema of those models, which may prove to be too generic for certain reporting needs;
- **Superficial requirements may refer to the choice of graphical notation**, since stakeholders primarily perceive models visually. They may prefer popular notations (e.g. BPMN) for a more efficient learning curve across the organization, or they may prefer domain-specific aspects to ornament the notation in ways that optimize Moody's visual variables (discriminability, dual coding, semiotic clarity etc.). The notion of "secondary notation" and its nudging effect on model comprehension have also been recognized by BPM literature (Petrușel et al., 2017). The notational requirements are driven by semantics – i.e. hiding or displaying some ornamentation is a decision based on machine-readable properties attached to that element, thus the metamodel is influenced by such requirements;
- **Business process modeling is a form of multi-perspective modeling**, where certain perspectives (e.g. the resource perspective, the document perspective) are sometimes expected to be described in more detail than the common pools/lanes or data objects. Certain perspectives (e.g. risks and their mitigation, KPI dependencies) may be entirely absent from popular languages, requiring new types of models and semantic links to connect them to the process description. Semantic links exist between elements of models of different types (e.g. conversation, collaboration, choreography) and those may have to be extended with annotation properties to capture case-specific aspects;
- **Business process modeling is a form of hierarchic modeling**, where processes must be detailed across multiple levels of detail, from high-level process landscapes or company maps to low-level workflows, or even click-level control flows aiming for Robotic Process Automation. Either top-level decision-making criteria or low-level automation needs will inform the process conceptualization enrichment for an adequate alignment across these levels of detail, with business process models acting as a mitigation layer, top-level model types being aligned with strategic decision-making features and low level model types being aligned with the execution environment or IT architecture;
- **Stakeholders may expect some level of usability or automation** (generation of model skeletons) to support the very act of modeling, where sufficient information is available for such features (e.g. recommending the possible next model element based on the metamodel). These can be classified as requirements pertaining to the "modeling procedure" building block.

Traditional classes of requirements also apply here – e. g. security/privacy requirements regarding model management; however, the categories exemplified above are those that directly impact the modeling language conceptualization, even if the overall BPM project relies on a popular language or method.

The Agile Modeling Method Engineering "Plug-in": Exemplary Application

Based on observations on both commercial product customizations and open research community projects on Conceptual Modeling (Bork et al., 2019), the Agile Modeling Method Engineering framework was crystallized from learned lessons and metamodeling best practices as a methodology for managing modeling method requirements and for rapid development of enterprise-specific BPM software. Thus AMME becomes a plug-in for BPM lifecycles, as a subprocess for handling the escalations triggered by change requests that lead to custom BPM methods, usually built around a core of model types that themselves emerged in the past as requested process extensions (Figure 4, right). AMME prescribes several phases with feedback loops as illustrated in Figure 4 (left) – each phase is supported by dedicated methodological or technological enablers, summarized in the following:

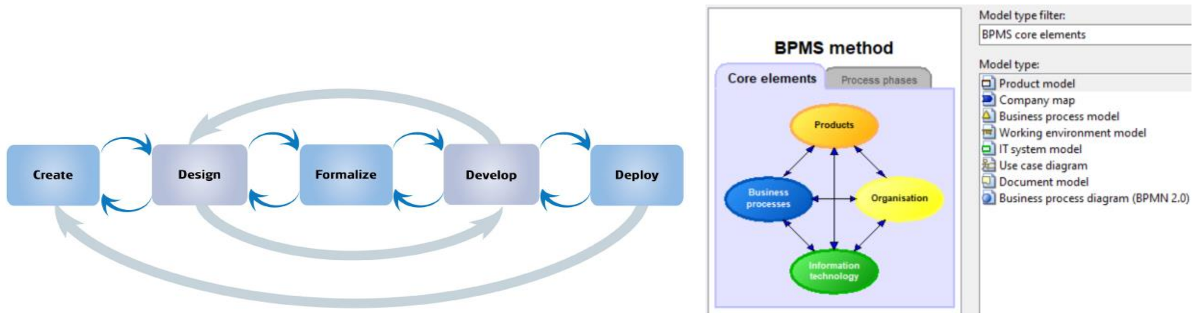


Figure 4. Left: the AMME subprocess (Karagiannis, 2016); Right: new types of models complementing BPMN

Create = elicitation of modeling method requirements and acquisition of domain knowledge. This phase benefits from the CoChaCo method for managing modeling method requirements (Karagiannis et al., 2019) – a meta-method that captures, with modeling means, the key concepts of a metamodeling project (goal, purpose, domain-specific conceptual hierarchies etc.). It is implemented in a requirements management tool that allows modeling concepts to be traced to modeling purposes, procedure, domain-specializations and design decisions regarding their visual manifestation in a BPM tool. Figure 5 shows samples of requirements models created with CoChaCo for a technology-specific customization of a process modeling tool, where the concept of Mobile App was a key resource for process execution.

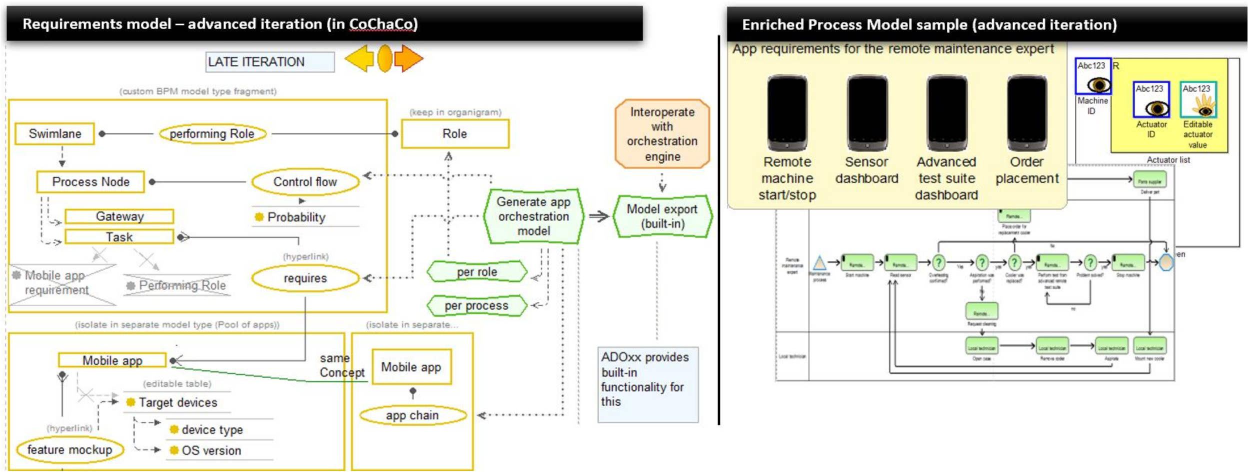


Figure 5. Model of "modeling method requirements" (left) and corresponding customization (right) - adapted from (Karagiannis et al., 2019)

Design = specification of the method building blocks. This phase produces specification documents of modeling method building blocks (notation, syntax, semantics, functionality, procedure) which can be partly generated from CoChaCo requirements models created in the preceding phase. A detailed understanding of change propagation paths across the specification building blocks is essential here to ensure a timely documentation of these changes and planning of the re-implementation effort. An inventory of reusable resources (notations and modeling scripts) allows for responsive redesigns and these specifications accumulate in a knowledge base and repository of reusable specification parts.

Formalize = formal specification of the method building blocks. This has scientific purposes, but is most often skipped in commercial projects where responsiveness to pragmatic customer needs gain priority.

Develop = realization of the modeling tool. This phase benefits from rapid protoyping metamodeling platforms, see ADOxx (BOC, 2020), whose meta-metamodel covers necessities identified over the history of adoptions of commercial process modeling tools. Later it was further extended with features of more experimental interest for open research problems – e.g. the ability to export any diagrammatic models to RDF graphs, opening a path for integration with the Knowledge Graph paradigm (Karagiannis and Buchmann, 2016), but not currently reflected in commercial deployments.

Deploy = deployment of the modeling tool. Customized modeling tools can be deployed either as desktop applications, mobile applications or cloud-based. These options are subject to contract-based infrastructure choices and are not really shifting between consecutive BPM lifecycle iterations.

It may be argued that modeling method requirements as described here are a rationale for the versioning of standards, and thus should be kept below the abstraction level of BPM lifecycles. However, it often happens that changes are expected from one lifecycle iteration to the next, within the same project, impacting the means of next iteration's process redesign/ analysis. Here is a list of exemplary requests, also reflected in Figure 6 across three classes of requirements (notational, semantic and functional):

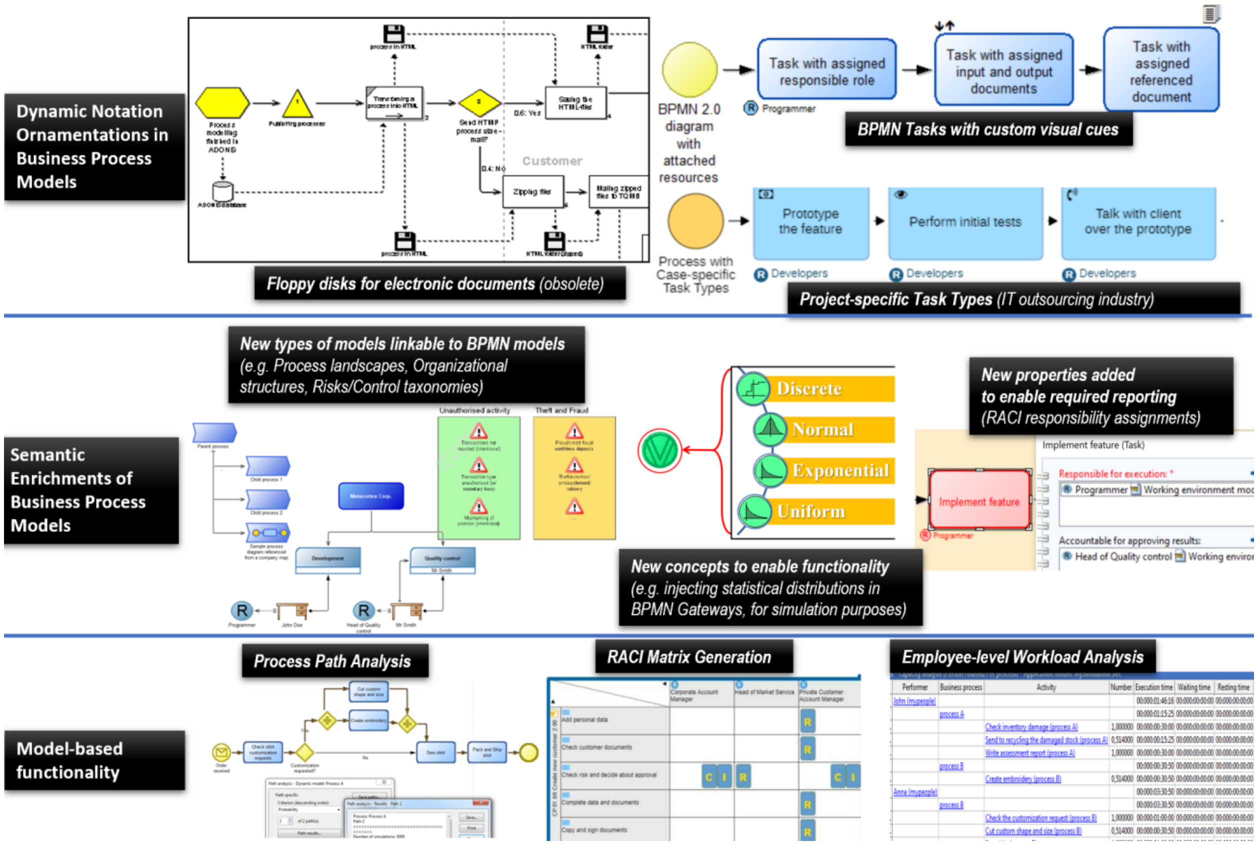

Dynamic Notation Ornamentations in Business Process Models
Floppy disks for electronic documents (obsolete)
BPMN 2.0 diagram with attached resources
Task with assigned responsible role
Task with assigned input and output documents
Task with assigned referenced document
Programmer
BPMN Tasks with custom visual cues
Process with Case-specific Task Types
Prototype the feature
Perform initial tests
Talk with client over the prototype
Developers
Project-specific Task Types (IT outsourcing industry)
Customer
Semantic Enrichments of Business Process Models
New types of models linkable to BPMN models (e.g. Process landscapes, Organizational structures, Risks/Control taxonomies)
Discrete
Normal
Exponential
Uniform
New properties added to enable required reporting (RACI responsibility assignments)
Implement feature (Task)
Implement feature
Responsible for execution:
Accountable for approving results:
Head of Quality control
New concepts to enable functionality (e.g. injecting statistical distributions in BPMN Gateways, for simulation purposes)
Model-based functionality
Process Path Analysis
RACI Matrix Generation
Employee-level Workload Analysis

Figure 6. Exemplary additive changes to product ADONIS (BOC, 2020b) motivated by case-specific modeling method requirements

- the customer would like to have a workload report per employee and department, thus needing a detailed organigram instead of the pools/lanes that typically represent process participants;
- an internal regulation forbids the assignment of employee instances in process models, requiring abstract roles to be used, and a complete absence of personal details (or of supply chain partners);
- the graphical notation must be adapted to some corporate culture element or should avoid some sensitive visual cue or an obsolete one (e.g. the floppy disk symbol for electronic documents);
- simulation features should be enhanced with a variety of statistical distributions for event probabilities;
- matrix views (e.g. RACI) should generated but require additional concepts or relations to be connected to the core process elements (e.g. more types of responsibilities than what pools/lanes can express);
- new types of models are required by stakeholders who want to take full benefit once they grasp the customization possibilities; these types of models often complement the business process models with additional perspectives – e. g. risk taxonomies and mitigation catalogues, product feature models, KPIs.

Table 1 suggests some exemplary quantitative efforts for customer projects in various industries/domains. Different types of requirements are exemplified and related to the efforts (over 2-3 iterations) to address those requirements.

Requirement Type	**Requirement Details**	**Industry**	**Effort** (person days over all iterations)
Notation customization	Define / adapt the standard notation of modelling constructs to provide explicit graphic cues for support elements	Telecom	Domain specification / design effort: 7. Implementation effort: 4 2 iterations
Metamodel extension	Extend the BPMN standard for governance/risk pertaining to the Sarbanes Oxley Act	Insurance	Domain specification/ design effort: 20 Implementation effort: 33 3 iterations
Semantic elevation for model processing	Support process maturity assessments in a distributed manner (data actuality, RACI organization assessment)	Production	Domain specification / design effort: 11 Implementation and testing effort: 27 2 iterations
Integration with external systems	Connect / re-use information artefacts stored externally and integrate with BPM environment for a Digital Twin	Transportation	Domain specification / design effort:8 Implementation effort: 13 3 iterations

Table 1. Exemplary requirements and corresponding efforts

Conclusions

The paper proposes a coupling of BPM lifecycles and AMME as a lifecycle plug-in whose application resulted in successful customer-specific customizations of BPM products through an agile metamodeling approach. The proposal originates from the observation that BPM lifecycles trigger escalation processes that reclaim adaptations of the process modeling method to a variety of requirements distinguishable according to a niche taxonomy. The BPM tool ADONIS, which is available in a community edition whose modeling language and model-based functionality can be consulted on-line (BOC, 2020b), shows an evolutionary snapshot resulting from this lifecycle extension, as the product assimilates selected new elements across all the building blocks of its modeling method - from one project to another, and also from one lifecycle iteration to another. The contribution is intended to inform on one hand BPM practitioners whose servitization packages may consider this in their service designs; on the other hand, it also informs researchers on lifecycle management and service co-creation about a specific manifestation of those paradigms in the BPM field. The literature on BPM lifecycles may further extrapolate how such conceptual artefacts are being transformed by advances in the BPM paradigm, as the "method agility requirement" hereby discussed may assimilate diverse sources of requirements (e.g. automation, process blockchains) and may inspire further opportunities for revisiting traditional BPM lifecycles.

REFERENCES

Aalst, W. van der et al. 2011. *Process Mining Manifesto*, available at https://www.win.tue.nl/ieeetfpm/downloads/Process%20Mining%20Manifesto.pdf

Bernardo, R., Ribeiro, G., and Dallavalle de Pádua, S. 2017. "The BPM lifecycle - How to incorporate a view external to the organization through dynamic capability," *Business Process Management Journal* (23:1), pp. 155 – 175.

Bizcon. 2019. RPA Lifecycle, available at https://bizcon.dk/our-solutions/rpa/.

BOC GmbH. 2020. The ADOxx Metamodeling Platform, available at https://www.adoxx.org.

BOC GmbH. 2020b. ADONIS Community Edition, available at https://www.adonis-community.com/en.

Bork, D., Buchmann, R., Karagiannis, D., Lee, M., Miron, E. T. 2019. "An open platform for modeling method conceptualization: The OMiLAB Digital Ecosystem," *Communications of the Association for Information Systems* (44), pp. 673-697

Buchmann, R. A., Ghiran, A. M., Döller, V., and Karagiannis, D. 2019. Conceptual Modeling education as a Design Problem. *Complex Systems Informatics and Modeling Quarterly* (21), pp.21-33.

Di Ciccio, C., Marrella, A., and Russo, A. 2015. "Knowledge-intensive Processes Characteristics, Requirements and Analysis of Contemporary Approaches," *Journal on Data Semantics* (4:1), pp. 29–57.

Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. 2016. *Fundamentals of Business Process Management*. Heidelberg: Springer, 2016.

Frank, U. 2013. "Domain-specific modelling languages: requirements analysis and design guidelines" in *Domain Engineering*, Berlin Heidelberg: Springer, pp. 133–157.

Gartner. 2020. *Gartner Glossary*, available at https://www.gartner.com/en/information-technology/glossary/business-process-management-bpm.

Gupta, D., and Prakash, N. 2001. "Engineering methods from method requirements specifications". *Requirements Engineering* (6), pp. 135–160.

Harkai, A., Cinpoeru, M., and Buchmann, R. A. 2018. "The What facet of the Zachman Framework – a Linked Data-driven interpretation", in *Proceedings of Workshops at th3 30th International Conference on Advanced Information Systems Engineering*, Tallinn, Estonia, pp. 197-208.

Henderson-Sellers, B., Ralyté, J., Âgerfalk, P., and Rossi, M. 2014. *Situational Method Engineering*, Berlin Heidelberg: Springer.

Kambil, A., Friesen, G.B., and Sundaram A. 1999. "Co-creation: A New Source of Value". *Accenture Outlook* (2), available at http://kambil.com/wp-content/uploads/PDF/accenture/cocreation2.pdf

Karagiannis, D. 2016. "Conceptual modelling methods: the AMME agile engineering approach", in *Proceedings of 15th International Conference on Informatics in Economy*, Cluj Napoca, Romania, pp. 3-19.

Karagiannis, D., and Buchmann, R. A. 2016. "Linked Open Models – extending Linked Open Data with conceptual model information," *Information Systems* (56) pp. 174-197.

Karagiannis, D., and Kühn, H. 2002. "Metamodelling Platforms", in *Proceedings of EC-Web 2002 – DEXA 2002*, Aix-en-Provence, France, pp. 182.

Karagiannis, D., Burzynski, P., Utz, W., and Buchmann, R. 2019. "A metamodeling approach to support the engineering of modeling method requirements,", in *Proceedings of the 27th IEEE Requirements Engineering Conference*, Jeju Island, Korea, pp. 199-210.

Karagiannis, D., Junginger, S., and Strobl, R. 1996. "Introduction to Business Process Management Systems Concepts", in *Business process modelling*, Berlin Heidelberg: Springer, pp. 81-106.

Karagiannis, D., Mayr, H. C., and Mylopoulos, J. (eds.) 2016. *Domain-specific Conceptual Modeling*, Basel: Springer.

Kelly, S., and Tolvanen, J.-P. 2008. *Domain-Specific Modeling: Enabling Full Code Generation*, New Jersey: John Wiley & Sons.

Kelly, S., Lyytinen, K., and Rossi, M. 2013. "MetaEdit+ a fully configurable multi-user and multi-tool CASE and CAME environment", in *Seminal Contributions to Information Systems Engineering*, Berlin Heidelberg: Springer, pp. 109–129.

Maier, R. 2004. *Knowledge Management Systems*, Berlin Heidelberg: Springer.

Moody, D. L., Heymans, P., and Matulevičius, R. 2009. "Improving the effectiveness of visual representations in requirements engineering: An evaluation of i* visual syntax", in *Proceedings of 17th IEEE Requirements Engineering Conference*, Atlanta, USA, pp. 171-180.

de Morais, R. N., Kazan, S., de Padua, S. I. D., Costa, A. L. 2014. "An analysis of BPM lifecycles: from a literature review to a framework proposal," *Business Process Management Journal* 20(3), pp. 412-432.

Petrușel, R., Mendling, J., and Reijers, H. 2017. "Task specific visual cues for improving process model understanding ," in *Information and Software Technology* (79) pp. 63-78.

PNM SOFT. 2017. BPM Lifecycle, available at http://www.pnmsoft.com/resources/bpm-tutorial/bpm-lifecycle/

Polancic, G. 2013. "Learning BPMN 2.0 – Business Process Vs Workflow," available at http://blog.goodelearning.com/bpmn/business-process-vsworkflow.

Pourshahid, A., Amyot, D., Peyton, L., Ghanavati, S., Chen, P., Weiss, M., and Forster, A. 2009. "Business process management with the user requirements notation," *Electronic Commerce Research* (9), pp. 269–316.

Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., and Bettin, J. (eds.) 2013. *Domain Engineering*, Berlin Heidelberg: Springer.

Szelagowski, M. 2018. "Evolution of the BPM lifecycle", in *Proceedings of the Federated Conference on Computer Science and Information Systems*, Poznan, Poland, pp. 205-211.

Vargo, S. L., and Lusch, R. F. 2004. "Evolving to a New Dominant Logic for Marketing", *Journal of Marketing* (68:1), pp. 1-17.

Wisniewski, D., Potoniec, J., Lawrynowicz, A., and Maria Keet, C. 2019. "Analysis of ontology competency questions and their formalizations in SPARQL-OWL", *Journal of Web Semantics* (59), https://doi.org/10.1016/j.websem.2019.100534