# Agile Modeling Method Engineering

Dimitris Karagiannis
Faculty of Computer Science, University of Vienna
Waehringerstr. 29, Vienna, A-1090, Austria
+43-1-4277-78910
dk@dke.univie.ac.at

## ABSTRACT

By repurposing agility principles established in software engineering, this paper provides an overview on the practice of **Agile Modeling Method Engineering** (AMME) driven by evolving requirements and motivated by emerging paradigms and research initiative – e.g., Enterprise Modeling, Factories of the Future, Internet of Things, Cyber-physical Systems. The approach has emerged from experiences with meta-modeling projects developed within the frame of the Open Model Initiative Laboratory (OMILab), where flexibility challenges have been raised by (a) evolving modeling requirements, (b) modeling requirements propagating from run-time systems requirements, as well as (c) requirements pertaining to domain-specificity. The framework and the characteristics of AMME are hereby discussed with respect to both methodological and architectural aspects and the Open Model Initiative Laboratory is be presented as an instance setup of the generalized AMME framework.

## Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**] Model Development – *modeling methodologies*

## General Terms

Design, Languages.

## Keywords

Agile modeling method engineering, metamodeling, model-aware information systems, conceptual modeling

## 1. INTRODUCTION

Diagrammatic modeling methods and languages are commonly perceived as stable, even standardized, artefacts that establish some commonly agreed way of describing a "system under study" with diagrammatic means. This implies that all stakeholders work on the same level of abstraction and specificity, and their needs are aligned with the high-level (cross-domain) requirements for which the modeling method/language was created. On the other hand, as it is revealed by the aphorism "all models are wrong, but only some are useful", modeling is fundamentally an abstraction effort - some properties are omitted, and only those that are considered "useful" are captured in the underlying conceptualization. The usefulness distinction - between the properties captured as "first-class modeling citizens" and those that are omitted - must be based on the **modeling requirements** of targeted stakeholders. For standardized modeling methods/languages, the "targeted stakeholders" are an abstract mass of potential users dealing with the same class of (recurring) problems. Consequently, standards provide artefacts that are highly reusable within that class of problems, typically regardless of the application domain. They establish a common level of abstraction, and domain specificity is sacrificed for the benefit of reusability across domains. The quality of modeling languages is typically discussed with respect to some internal, "inwards" quality criteria – e.g., consistency, integrity, performance (often for some implementation and not for the language itself). Modeling requirements are obscured in the high-level class of addressed problems, and the standard may be later confronted with practical experience and feedback [1] that is difficult to assimilate in a timely manner, as the evolution and versioning of standards is rather slow. There are also situations where stakeholders have a more identifiable and involved presence, their modeling requirements are traceable and evolving, or there is an inherent abstraction variety in the addressed class of problems (e.g., a requirement for multi-level domain specificity [2]) – all these are characteristics that reclaim an **Agile Modeling Method Engineering** (AMME) approach. Under such conditions, the quality criteria for modeling languages shift to "outwards" considerations – usefulness (relative to traceable requirements), comprehensibility/end-user acceptance, usability (relative to a specific implementation).

PCI 2015, October 01 - 03, 2015, Athens, Greece
Copyright is held by the owner/author(s).
Publication rights licensed to ACM.
ACM 978-1-4503-3551-5/15/10…$15.00
DOI: http://dx.doi.org/10.1145/2801948.2802040

The goal of this paper is to outline the key characteristics of AMME as an emerging paradigm for tackling evolving modeling requirements emerging from a narrow domain and for specific needs of modeling stakeholders. The remainder of the paper is structured as follows: Section 2 provides the motivation frame, by relating the notion of agility to the enterprise modeling context. Section 3 outlines the characteristics of AMME and draws a generic framework to guide its deployment. Section 4 presents the Open Model Initiative Laboratory [3] as an instance setup of AMME, with feasibility already evaluated in existing or on-going meta-modeling projects. The paper ends with references to related works, followed by a SWOT analysis to highlights the benefits and limitations of the current maturity level for AMME.

## 2. AGILITY: A REQUIREMENT IN ENTERPRISE MODELING

Although not limited to this, the relevance of AMME can be best highlighted and motivated in the context of the synergy between the Enterprise Modeling paradigm and recent industry-oriented research initiatives and roadmaps - e.g., the global movement of Internet of Things, USA's initiative on Cyber-Physical Systems [4], Japan's new Robot Strategy [5], the European Factories of the Future [6]. Enterprise Modeling has produced a rich variety of established enterprise architecture frameworks and modeling methods – e.g., Archimate [7], Zachman [8], EKD [9], capability-oriented approaches [10]. Variation manifests not only in the levels of abstraction, but also in the multitude of perspectives they provide on the enterprise, relative to some envisioned modeling goals. It can also be the case that an enterprise modeling method is required for a rather narrow application domain or even for a single enterprise case, where reusability outside the enterprise context is sacrificed to the benefits of case-specific semantics and familiarity of stakeholders with the modeling context. In order to establish the **motivational frame** for AMME, this section compiles some key enterprise characteristics from the aforementioned research roadmaps, as well as from project experience accumulated through the Open Model Initiative Laboratory [3] (an instance setup of AMME - see Section 4):

**Complexity.** The "system under study" has an inherent complexity that comprises multiple views/facets/perspectives to be captured on modeling level, demanding a decomposition approach to track modeling requirements in relation to modeling method building blocks. The alphabet of the modeling language must be partitioned into subsets addressing different facets of the modelled enterprise, while preserving consistency. The method engineer is called to make decomposition decisions which will also map on the granularity of backlog items to be managed during AMME.

**Changing requirements.** Enterprise modeling requirements are fundamentally unstable and evolving due to factors such as: (a) the method engineer will develop a gradual understanding of the application area; (b) the level of domain-specificity will not be necessarily fixed, as the engineer may be confronted with a case of multi-level modeling (in the sense of [2]) or with a gradual need to deepen concept specialization; (c) the stakeholder will require "throwaway prototypes" to get an early idea of what the modeling method can provide.

**"Conceptual model"-awareness at run-time**. The behavior of a run-time information system or devices is parameterized and influenced by information retrieved from models, making them "model-aware". Consequently, run-time requirements propagate in modeling requirements. The method engineer must make sure that the run-time systems will find sufficient (machine-readable) richness and granularity in the semantics exposed by the language.

**Modeling as knowledge representation**. Modeling is not employed for documentation or for communicating with some external bodies (e.g., partners, auditors). Instead, it provides knowledge representation means for internal purposes (including the aforementioned model-awareness in run-time systems). This is an evolution from earlier approaches where modeling was employed to the aim of code generation or for configuring the behavior of existing systems (e.g., in process-aware information systems). In enterprise modeling, models are means of representing the relevant knowledge pertaining to different facets (e.g., processes, resources, business context, business model) and the metamodel becomes a terminological box (Tbox) resulted from an iterative knowledge acquisition effort.

In the context hereby outlined, AMME is called to ensure that timely solutions and response to changes in modeling requirements will be provided, just like the Agile Manifesto [11] aimed to overcome the shortcomings of rigid software development approaches (e.g., waterfall).

## 3. THE AMME METHODOLOGY

### 3.1 Propagation and Evolution of Modeling Requirements

The Agile Manifesto [11] was a pragmatic reaction to rigid software development methodologies and managed to establish management approaches built on the following pillars: (i) *Iterative development* (work cycles allow for revisiting the same work items); (ii) *Incremental development* (successive usable versions are built upon previous versions); (iii) *Version control* (the output of each iteration is traceable across multiple versions and in relation to their requirements); (iv) *Team control* (small groups of people are assigned to backlog items with shared accountability).

In the practice of AMME, the agility principles are applied on the fundamentals of modeling method design, with a backlog granularity refinement that starts from the method building blocks defined by [12], which in turn determine different classes of modeling requirements.

Table 1 describes briefly these building blocks and maps them on typical cases of requirements propagation (when evolution in a class of requirements triggers indirect changes in another class).

Further on, Fig. 1 illustrates the notion of **evolving modeling requirements at semantics level**, exemplified on the common concept of Activity/Task that can be found in most workflow, business process or enterprise modeling methods. The evolution is exemplified along three stages: *Stage 1*: the concept is generic in nature, being expressed strictly through its assigned visual symbol. The semantics are established at an informal level (in the modeling guidelines), being largely left to human interpretation – such an activity can be used to designate a high level supply chain phase, a low-level work task or a data processing step in some algorithm. *Stage 2*: the concept semantics are enriched by subtyping, narrowing down and visually distinguishing possible interpretations (manual task, automated task etc.). *Stage 3*: the concept is further enriched with a set of definitorial and machine-readable properties that are prescribed by the method engineering (in the language semantics). Now we have a business activity, defined as "something" that takes execution time, has associated cost, responsibility and resource allocation etc. Outside an AMME strategy, these three levels are provided by three different modeling languages supporting different levels of domain specificity (the samples in the figure are actually designed with UML, BPMN and ADONIS:CE [13], respectively). During an AMME process, these are three evolutionary stages that must be tracked for the same concept, in the same modeling method, while responding to evolving requirements pertaining to the depth of concept specialization and domain-specificity.

Fig. 2 illustrates a similar argument with respect to the modeling notation. Standards typically employ symbols aiming for simplicity or based on historical reasons. For example, the rhombus shape commonly used to depict decisions/gateways can be traced back to the earliest representations of flow charts drawn with crayon and template rulers in the field of mechanical engineering (the origin of flow charts is attributed to [14]). However, template rulers were not created to express information through visual shapes, but rather to distinguish between different concepts by assigning them different shapes. Familiarity is only derived from tradition and standardization and not from their cognitive value.

**Evolving notational requirements** may lead to the adoption of "customized notations" (Stage 3) at a level of the targeted enterprise or, even further, to notational requirements that propagate into semantics. The Stage 4 sample in the figure shows the symbol of an activity as defined the domain-specific project-based ComVantage modeling method [15]. It is enriched with interactive hotspots (hyperlinks) for navigating to related models of various types, along some semantic hyperlinks – to a model describing requirements for mobile app support, a model depicting a subprocess and an organizational model where the responsible role/employee can be found. The navigability implementation also triggers an enrichment of the concept semantics, by ensuring that the concept property set includes machine-readable references to modeling elements of specific types, from other models, and this must be assimilated in the language metamodel – therefore a usability requirement propagates to a requirement pertaining to semantics.

The AMME framework to be outlined in the next section is proposed by the work at hand to support modeling method engineering driven by the hereby discussed evolution and propagation for modeling requirements.

**Table 1. Requirements Propagation Concerns Mapped on Modeling Method Building Blocks**

| Modeling Method Building Block | Modeling Requirements Propagation Cases |
|---|---|
| The **modeling language** establishes the language alphabet (notation, grammar and machine-readable semantics for each concept and relation). It also partitions the alphabet in model types to achieve a manageable granularity and also to separate concerns and to ensure model comprehensibility for the modeler. | • Integration between model types may be achieved either at language level (semantic relations acting as hyperlinks between different models) or at functionality level (algorithms for consistency checks or consistency preservation).<br>• Modeling requirements may be imposed on notation, with respect to notation interactivity. This will propagate towards semantics (e.g., dynamic changes in the notations determined by the presence of some machine-readable properties).<br>• Requirements will propagate between the backlog items of the modeling language, for example a single annotation property may evolve into a concept with additional properties, and a concept may evolve into a new model type describing it in diagrammatic structures. |
| The **mechanisms and algorithms** cover functionality that will process the model information in order to satisfy various functional requirements (e.g., evaluation for decision support, model transformation, model integrity checks, interoperability with run-time systems). | • Requirements will propagate from model-aware run-time systems (regarding some pre-processing and serialization of the model information requested at run-time) and will further propagate towards the modeling language, to ensure that the requested properties are available in the models.<br>• Modeling requirements may also refer to design-time functionality which must be implemented to support the modelers in their decisions and evaluations (e.g., process simulation).<br>• Both types of requirements might further propagate to the modeling language, raising requests for the existence of additional concepts and semantics (properties, relations). |
| The **modeling procedure** defines the steps that must be taken by modelers towards some modeling goal (in the simplest case, it advises on the precedence required to create a set of models of different types). | • Modeling requirements may be present at this level in the form of non-functional constraints (e.g., "I don't want to deal with more than x model types"), which typically prove to be subject to very volatile changes (as the stakeholder becomes accustomed to the method).<br>• A more relevant case is when there is a requirement to automate some modeling procedure steps ("model of type x should be generated automatically") and this of course propagates to the mechanisms building block. |

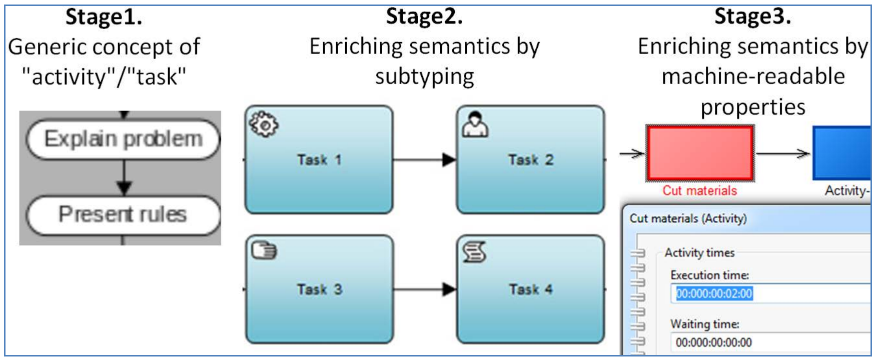

**Fig. 1. Concept evolution on semantic level**

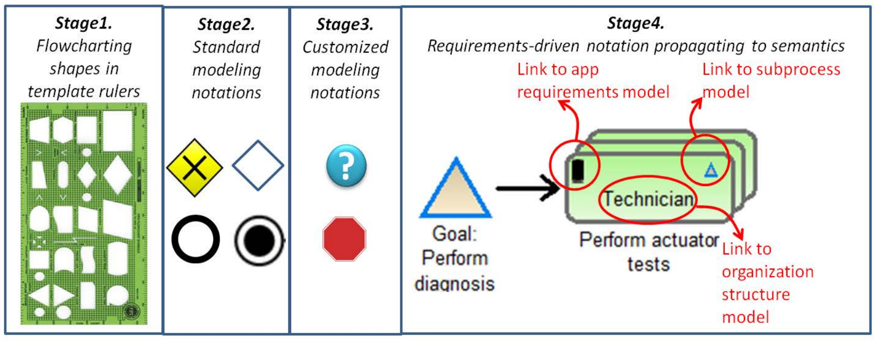
Stage1.
Flowcharting shapes in template rulers
Stage2.
Standard modeling notations
Stage3.
Customized modeling notations
Stage4.
Requirements-driven notation propagating to semantics
Link to app requirements model
Link to subprocess model
Technician
Goal: Perform diagnosis
Perform actuator tests
Link to organization structure model

**Fig. 2. Modeling evolution: the notation aspect**

## 3.2 AMME Framework and Architecture

The work at hand introduces AMME through its definitorial characteristics in response to changing requirements:

- ADAPTABILITY: The ability to modify existing concepts;
- EXTENSIBILITY: The ability to add new concepts;
- INTEGRABILITY: The ability to add bridging concepts in order to integrate existing ones;
- OPERABILITY: The ability to provide functionality for operating on models (e.g., simulation, transformation);
- USABILITY: The ability to provide satisfying user interaction and model understandability.

Agility in AMME relies the high level framework depicted in Fig. 3, which is driven by the iterative incremental cycle designated as the *Produce-Use* cycle with two high-level phases.

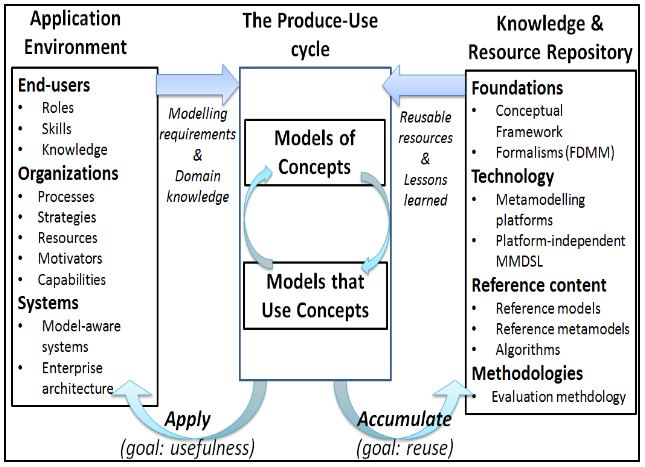

**Fig. 3. The AMME Framework**

In the top phase, the modeling language is defined by **Models of Concepts** - a working term that designates any kind of model having an ontological scope (i.e, aimed to describe categories of being and their relations). In the lower phase, once an iteration of the modeling language is provided to end-users, they are enabled to instantiate its constructs in order to create and evaluate **Models that Use Concepts** - we include here the actual enterprise models and any kind of model that instantiates the previously established domain understanding for the modelled case. The cycle is fed with modeling requirements from the "Application Environment", as well as the domain knowledge needed for domain conceptualization. In the back-end the "Knowledge and Resource Repository" accumulates reusable resources and lessons learned from different iterations, as well as from previous AMME projects to increase the productivity of future projects.

The framework must be supported by an architecture of methodological, conceptual and technological enablers, as depicted in Fig. 4, with the following modules:

The **AMME Project Tracking System** providing (a) the tracking of modeling requirements classified by the building blocks they refer to; (b) the tracking of backlog items mapped on similar building blocks, with an emphasis in propagation and integration tasks that must be re-evaluated as new requirements are accepted.

The **Reusable Asset Repositories** must ensure productivity and reusability in subsequent projects by accumulating granular assets that can be easily adapted and adopted in new modeling methods. A part of this is the documentation repository will include method specifications, the modeling procedure in form of end-user guidelines and a repository of assets used in iterative evaluations (model samples, questionnaires to assess model comprehension, previous evaluation results etc.).

The **Prototyping Environment** must support rapid prototyping, necessary for incremental development and throwaway prototypes. Metamodeling platforms are required here for increased productivity, possibly extended by some platform-independent technology for modeling method definition across multiple platforms at the same time. An instance of such technology will be highlighted in Section 4.

The **Deployment Channels** must provide ways to provide stakeholders with access to the incremental modeling prototypes, with different available options – software as a service, standalone installation or remote hosted installation. In addition, interoperability mechanism must expose models in machine-readable format required by model-aware run-time systems. Finally, dissemination channels must support both design-time and run-time stakeholders with documentation and reference content derived from the Asset Repositories.

## 4. AMME WITHIN OMILAB

The Open Model Initiative Laboratory [3] is a physical and virtual research environment that instantiates the envisioned AMME framework and architecture as support for meta-modeling research projects and communities. An early stage feasibility study for the Open Models Initiative was published in [16]. Some specifics pertaining to the OMILab setup are provided in the following:

The **OMILab lifecycle** defines the internal cycle of a method iteration, with several phases depicted in **Fig. 5**: (a) The *Creation* phase is a mix of knowledge acquisition and requirements elicitation activities that capture and represent the modeling requirements; (b) the Design phase specifies the metamodel, language grammar, notation and functionality; (c) the *Formalize* phase aim to describe the outcome of the previous phase in non-ambiguous representations with the purpose of sharing results within a scientific community; (d) the *Develop* phase produces concrete modeling prototypes; (e) the *Deploy/Validate* phase involves the stakeholders in hands-on experience and the evaluation process for the current iteration.

The **MM-DSL** is a platform-independent declarative language that allows code-based editing of modeling method definitions and their compilation for the *metamodeling platform* of choice. The language grammar is openly available at [17] and additional details are published in [18]. It relies on a meta-metamodel that is a common abstraction of the typical meta-metamodels provided by the popular metamodeling platforms. Currently a proof-of-concept compiler is available only for the metamodeling platform provided through OMILab, that is **ADOxx** [19] – a platform with multiple deployment options (cloud, service APIs, standalone installation). Additional compilers are expected to emerge from community-based efforts.

The **OMILab project repository** [20] acts as an Asset Repository and as Deployment Channel. It has accumulated a diversity of enterprise metamodeling projects in different stages of maturation and with various degrees of domain-specificity. Examples include the ComVantage method [15] (for collaborative business processes with requirements for mobile apps and Linked Data), CIDOC (for cultural heritage modeling), EKD, PROMOTE etc.

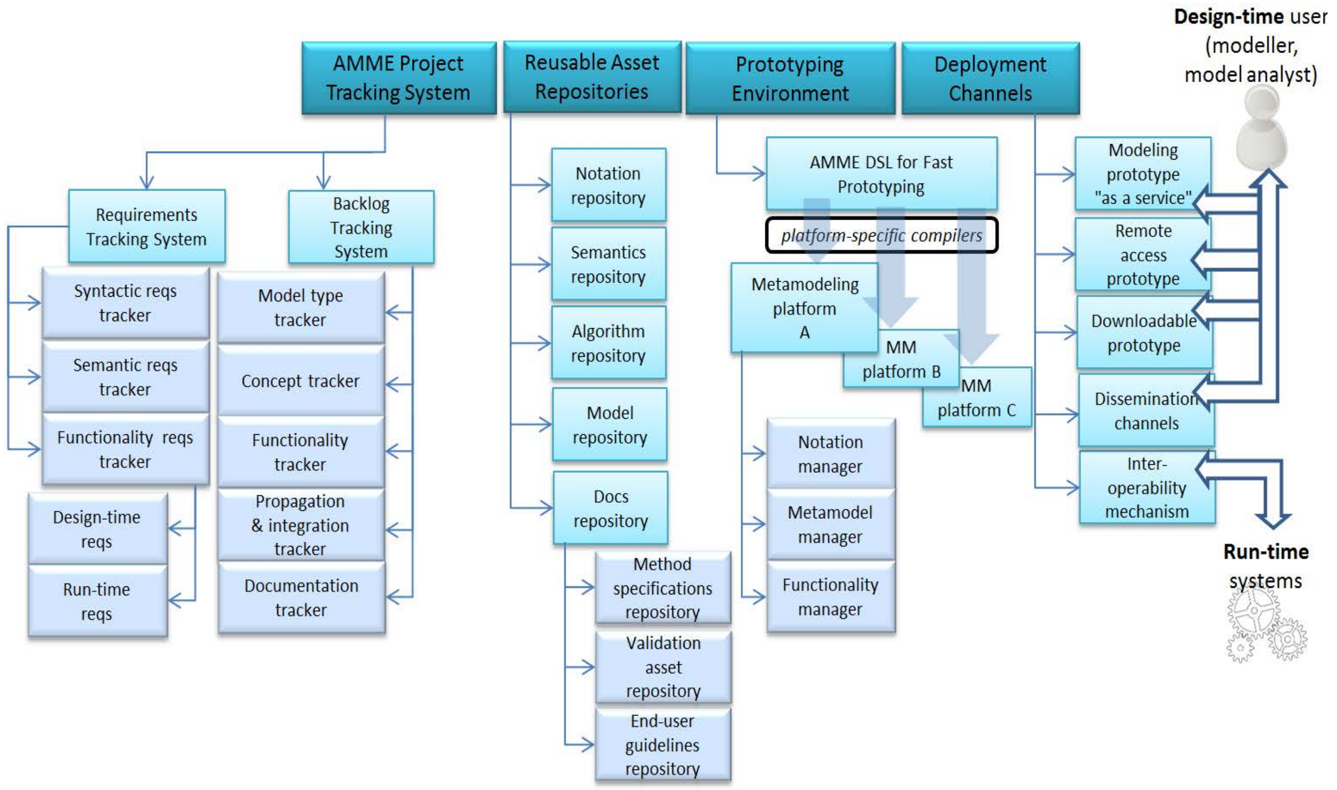

**Fig. 4. Support architecture for setting up an AMME approach**

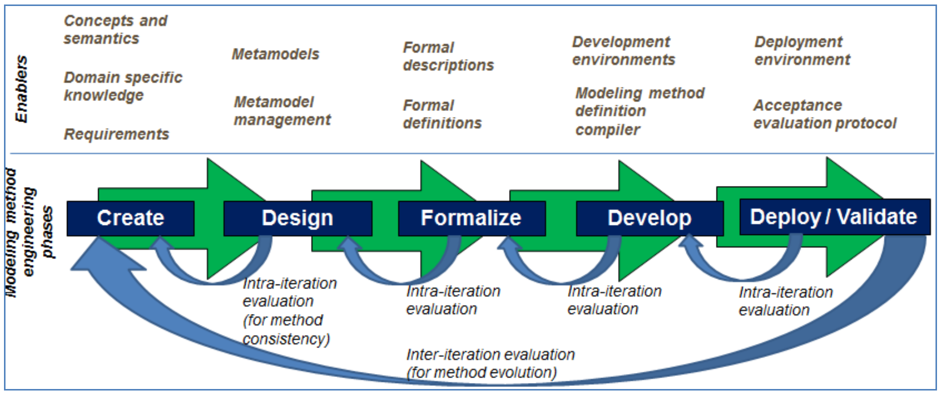
Enablers
Concepts and semantics
Domain specific knowledge
Requirements
Metamodels
Metamodel management
Formal descriptions
Formal definitions
Development environments
Modeling method definition compiler
Deployment environment
Acceptance evaluation protocol
Modeling method engineering phases
Create
Design
Formalize
Develop
Deploy / Validate
Intra-iteration evaluation (for method consistency)
Intra-iteration evaluation
Intra-iteration evaluation
Intra-iteration evaluation
Inter-iteration evaluation (for method evolution)

**Fig. 5. The iterative OMILab lifecycle**

## 5. RELATED WORKS

To the best of our knowledge, this is the initial explicit presentation of AMME, with some characteristics being previously suggested in the context of fast prototyping in [18]. Related works that converge in the hereby proposed notion come from the areas of model quality [21] and notation quality [22], since agility does not tackle only new modeling requirements, but also feedback for improvements with respect to existing requirements. Agility challenges pertaining to modeling methods can also be derived from the context of domain-specific multi-level modeling [2], as well as from Language-oriented Programming (see a metamodeling interpretation in [23]). Metamodeling itself has emerged from the need to enable flexibility across a multi-abstraction layered architecture (e.g., the MOF framework [24], tools such as [25]) - however, it has not been complemented yet by a methodological and architectural approach that repurposes the agility principles of software development. Project-based experiences are called to further provide methodological, managerial or technological enablers, or to report on best practices and pitfalls of agility in modeling method engineering.

## 6. CONCLUSIVE SWOT ANALYSIS

The necessity of AMME, just like the Agile Manifesto in context of software development, emerged from analyzing pragmatic challenges in concrete projects that could not be satisfied with more rigid, waterfall-style approaches. Therefore OMILab was introduced here both an instance deployment of AMME, as well as a research environment and corpus that helped consolidate the AMME vision by generalizing lessons learned from practical projects. Consequently, the generic AMME framework and its OMILab deployment participate to a cycle of mutual maturation.

A SWOT analysis is hereby provided to highlight the benefits and limitations of the current maturation level:

**Strengths:** AMME is based on a metamodeling strategy aimed to enabling and benefiting from the evolution of modeling requirements coming from different types of stakeholders - from design-time decision makers (aiming for model analysis) to run-time system users (relying on machine-readable model semantics). **Weaknesses:** Our experience with applying AMME is currently focused on OMILab projects developed by small research teams. Just like in the case of agile software engineering, community-driven practices and pitfall warnings must be shared to deploy AMME as a general practice. **Opportunities:** Future work may be layered on the conceptual foundation established by the work at hand, in order to enrich AMME as a community-driven framework, to raise its maturity level and to enrich its tool support with respect to the different modules of the proposed architecture – management support tools, versioning control and traceability strategies etc. **Threats:** Stakeholders may opt to adjust their needs to the features available in standards, to avoid the overhead of modeling method customization and the incremental learning effort associated to it. Awareness on the practice of AMME must therefore be raised to encourage the formulation of modeling requirements towards a new generation of enterprise modeling methods.

Consequently, we formulate the takeaway message: *Modeling requirements should be the essential driver for modeling method engineering, and an approach based on AMME will enable agile response to evolving requirements, as well as traceability of change propagation across modeling method building blocks.*

## 7. REFERENCES

[1]. Burlton, R. 2009. Perspectives on Process Modeling. BPTrends, http://www.bptrends.com/publicationfiles/07-09-COL-POV-Perspectives%20on%20Process%20Modeling-Burlton-cap%20_1_%20RB%20Final.pdf

[2]. Frank, U. Multilevel Modeling. In *Business & Information Systems Engineering*: 6(6), Springer, 319-337, DOI= 10.1007/s12599-014-0350-4

[3]. Open Model Initiative Laboratory, http://omilab.org

[4]. USA National Science Foundation, Cyber-physical Systems, http://www.nsf.gov/news/special_reports/cyber-physical/

[5]. Headquarters for Japan's Economic Revitalization. 2015. Japan's Robot Strategy – Vision, Strategy, Action Plan, www.meti.go.jp/english/press/2015/pdf/0123_01b.pdf

[6]. European Factories of the Future Association, Factories of the Future Roadmap, http://www.effra.eu/attachments/article/129/Factories%20of%20the%20Future%202020%20Roadmap.pdf

[7]. The Open Group, ArchiMate® 2.1 Specification, http://www.opengroup.org/archimate/

[8]. Zachman, J.A. 1987. A framework for information systems architecture. *IBM systems journal* 26(3), IBM, 276-292, DOI= DOI= 10.1147/sj.263.0276

[9]. Loucopoulos, P., Kavakli, V. 1999. Enterprise Knowledge Management and Conceptual Modelling. In: Goos, G., Hartmanis, J., van Leeuwen, J., Chen, P., Akoka, J., Kangassalu, H., Thalheim, B. (eds.): *Conceptual Modeling: Current Issues and Future Directions*, LNCS 1565, Springer, 123-143, DOI= 10.1007/3-540-48854-5_11

[10]. Zdravkovic, J., Stirna, J., Kuhr, J. C., Koc., H. 2014. Requirements Engineering for Capability Driven Development. In: Frank, U., Loucopoulos, P., Pastor, O., Petrounias, I (eds.): *Proceedings of POEM 2014*, LNBIP 197, Springer, 193-207, DOI= 10.1007/978-3-662-45501-2_14

[11]. Manifesto for Agile Software Development, http://agilemanifesto.org/

[12]. Karagiannis, D., Kühn, H. 2002. Metamodeling Platforms, In: Bauknecht, K., Min Tjoa, A., Quirchmayr, G. (eds.), *Proceedings of EC-Web 2002 – Dexa 2002*, LNCS 2455, Springer, 451-464, DOI= 10.1007/3-540-45705-4_19

[13]. BOC-Group, ADONIS Community Edition, http://en.adonis-community.com/

[14]. Gilbreth, F. B., Gilbreth, L. M. 1921. Process Charts. American Society of Mechanical Engineers, https://engineering.purdue.edu/IE/GilbrethLibrary/gilbrethproject/processcharts.pdf

[15]. Open Model Initiative Laboratory, ComVantage modelling prototype and resources, http://www.omilab.org/web/comvantage/home

[16]. Karagiannis, D., Grossmann, W., Hoefferer, P. 2008. Open Model Initiative – a Feasability Study, http://cms.dke.univie.ac.at/uploads/media/Open_Models_Feasibility_Study_SEPT_2008.pdf

[17]. The MM-DSL language specification, http://www.omilab.org/c/document_library/get_file?uuid=eb040aac-ea0d-4df7-a0a9-80b73f00c5f8&groupId=10122

[18]. Visic, N., Fill, H.-G., Buchmann, R., Karagiannis, D. 2015. A domain-specific language for modelling method definition: from requirements to grammar. In: Rolland, C., Anagnostopoulos, D., Loucopoulos, P., Gonzalez-Perez, C. (eds.), *Proceedings of RCIS 2015*, IEEE, 286-297, DOI= 10.1109/RCIS.2015.7128889

[19]. BOC-Group, ADOxx, http://www.adoxx.org/live/

[20]. Open Model Initiative Laboratory, Project Repository, http://www.omilab.org/web/guest/projects

[21]. Krogstie, J. 2012. Quality of Business Process Models. In: Sandkuhl, K., Seigerroth, U., Stirna, J. (eds.), *Proceedings of POEM 2012*, LNBIP 134, Springer, 76-90, DOI= 10.1007/978-3-642-34549-4_6

[22]. Moody, L. 2009. The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering* 35(6), IEEE, 756-779, DOI= 10.1109/TSE.2009.67

[23]. Clark, T., Sammut, P., Willans, J., Applied metamodelling: a foundation for language driven development, http://eprints.mdx.ac.uk/6060/.

[24]. Object Management Group, MetaObject Facility, http://www.omg.org/mof/

[25]. Kelly, S., Lyytinen, K., Rossi, M. 2013. MetaEdit+ a Fully Configurable Multi-user and Multi-tool CASE and CAME Environment, In: Bubenko, J., Krogstie, J., Pastor, O., Pernici, B., Rolland, C., Solvberg, A (eds.), *Seminal Contributions to Information Systems Engineering*, Springer, 109-129, DOI= 10.1007/978-3-642-36926-1_9