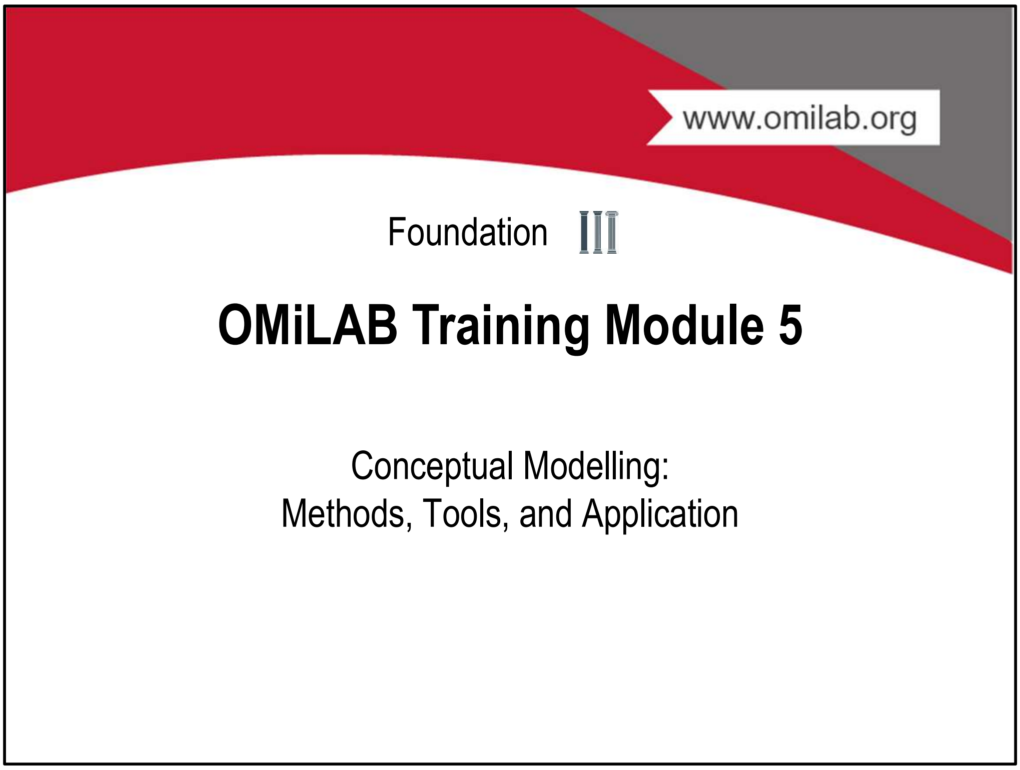
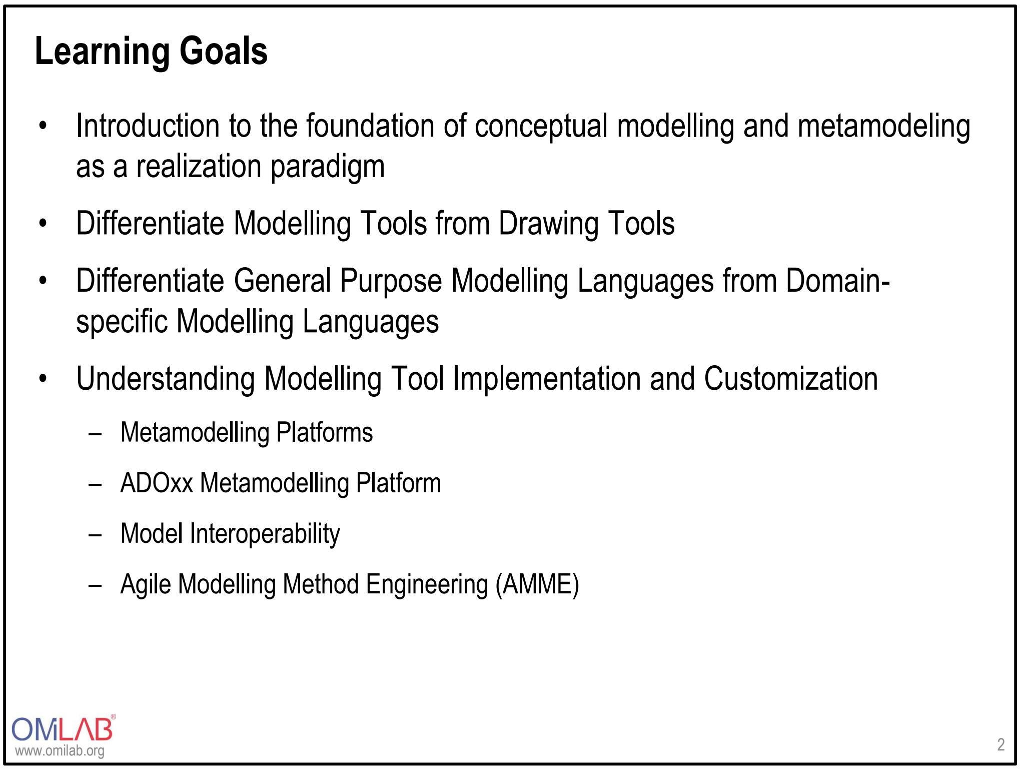


www.omilab.org

Foundation

OMiLAB Training Module 5

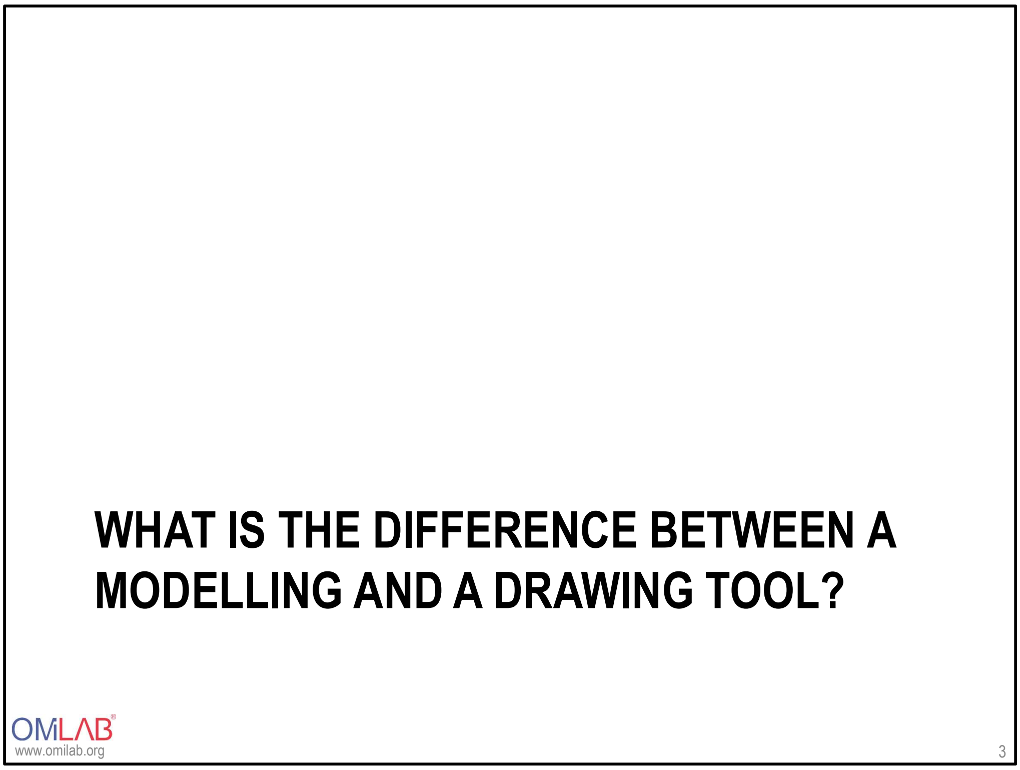
Conceptual Modelling:
Methods, Tools, and Application



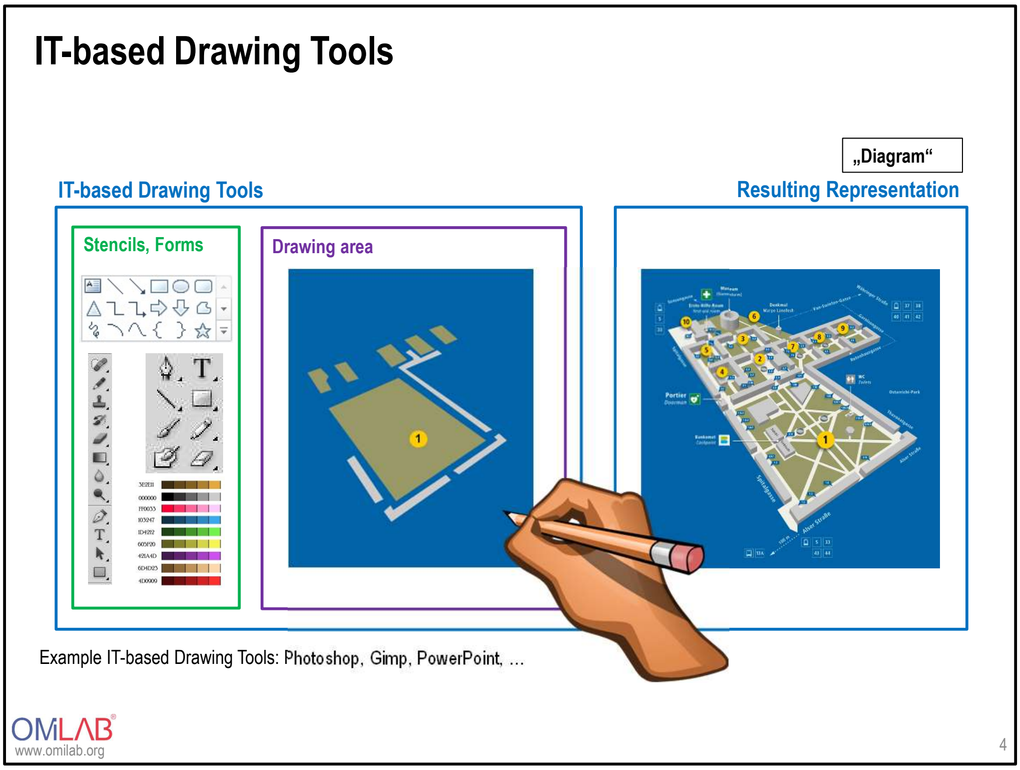
Learning Goals

- Introduction to the foundation of conceptual modelling and metamodeling as a realization paradigm
- Differentiate Modelling Tools from Drawing Tools
- Differentiate General Purpose Modelling Languages from Domain-specific Modelling Languages
- Understanding Modelling Tool Implementation and Customization
  - Metamodelling Platforms
  - ADOxx Metamodelling Platform
  - Model Interoperability
  - Agile Modelling Method Engineering (AMME)

- The goal of this module is to provide a proper understanding of the foundations of conceptual modelling methods, tools, and applications.
- Three major aspects are covered:
  - 1. What is the difference between a modelling tool and a drawing tool
  - 2. What is the conceptualization of a modelling method and what methods are available to help in the conceptualization?
  - 3. What kind of operations can be added to and executing in combination with a conceptual modelling tool.



WHAT IS THE DIFFERENCE BETWEEN A MODELLING AND A DRAWING TOOL?



IT-based Drawing Tools

„Diagram“

IT-based Drawing Tools

Resulting Representation

Stencils, Forms

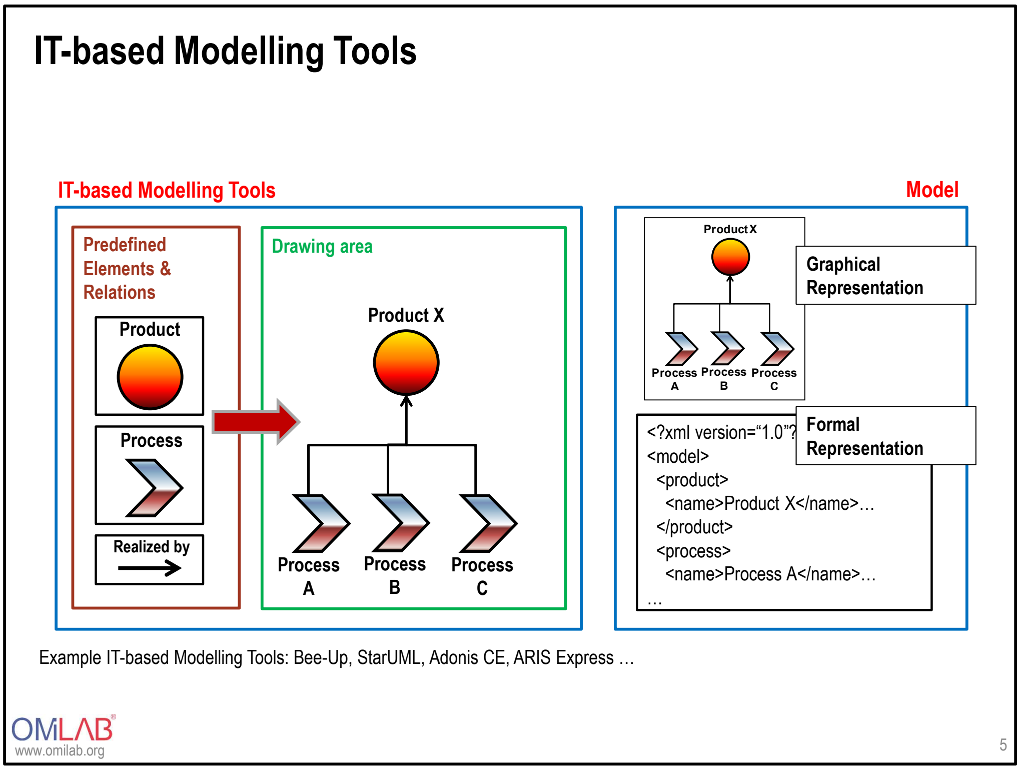
Drawing area

Example IT-based Drawing Tools: Photoshop, Gimp, PowerPoint, …

OMiLAB
www.omilab.org

4

- This slide shows a conventional IT-based drawing tool like PowerPoint and Gimp
- When using such tools, one can easily create drawings which look are nice looking



IT-based Modelling Tools

IT-based Modelling Tools

Predefined Elements & Relations

Product

Process

Realized by

Drawing area

Product X

Process A

Process B

Process C

Model

Product X

Process A

Process B

Process C

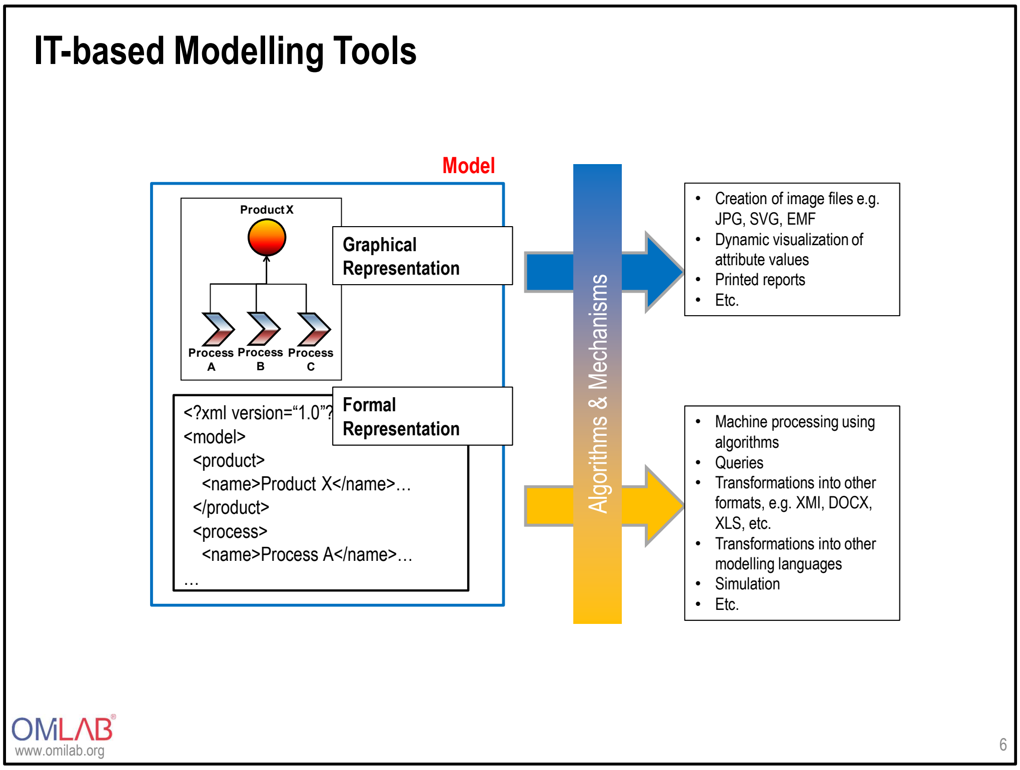
Graphical Representation

Formal Representation

```
<?xml version="1.0"?
<model>
  <product>
    <name>Product X</name>…
  </product>
  <process>
    <name>Process A</name>…
…
```

Example IT-based Modelling Tools: Bee-Up, StarUML, Adonis CE, ARIS Express …

- When now moving toward IT-based Modelling Tools, we can surely also create nice drawings
- However, now we have, aside from the graphical representation, also a formal representation
- The modelling tools does not store a product as a colored circle but as an instance of the concept product of a modelling language



IT-based Modelling Tools

Model

Graphical Representation

Formal Representation

```
<?xml version="1.0"?
<model>
  <product>
    <name>Product X</name>…
  </product>
  <process>
    <name>Process A</name>…
…
```

Algorithms & Mechanisms

- Creation of image files e.g. JPG, SVG, EMF
- Dynamic visualization of attribute values
- Printed reports
- Etc.

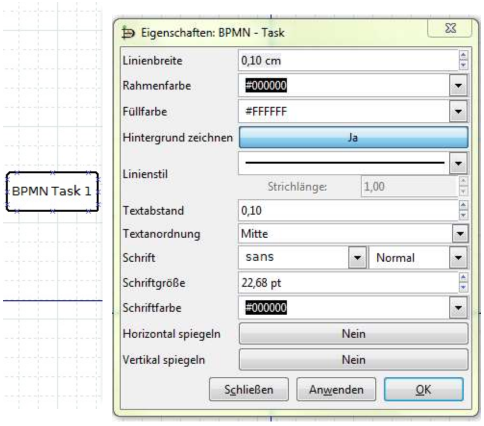
- Machine processing using algorithms
- Queries
- Transformations into other formats, e.g. XMI, DOCX, XLS, etc.
- Transformations into other modelling languages
- Simulation
- Etc.

OMILAB®
www.omilab.org

- Because we are using a Modelling Tool we can not only look at and print the models/diagrams
- We can also apply mechanisms & algorithms on them as they have all the modelling language information encoded in the formal representation
- Such processing of model information would not be possible with drawing tools like PowerPoint

Drawing- vs. Modelling Tools – Example (1)

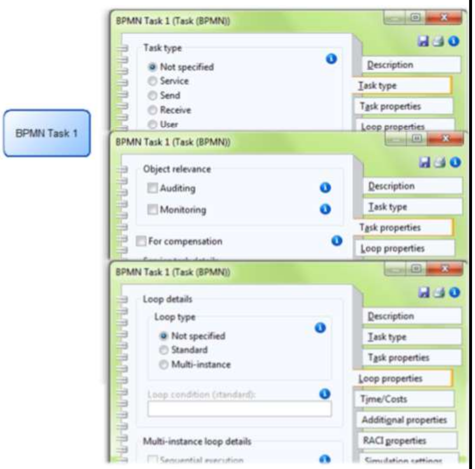
- Drawing Tool



Eigenschaften: BPMN - Task

Linienbreite	0,10 cm
Rahmenfarbe	#000000
Füllfarbe	#FFFFFF
Hintergrund zeichnen	Ja
Linienstil	Strichlänge: 1,00
Textabstand	0,10
Textanordnung	Mitte
Schrift	sans Normal
Schriftgröße	22,68 pt
Schriftfarbe	#000000
Horizontal spiegeln	Nein
Vertikal spiegeln	Nein

- Modelling Tool



BPMN Task 1 (Task (BPMN))

Task type
● Not specified
○ Service
○ Send
○ Receive
○ User

BPMN Task 1 (Task (BPMN))

Object relevance
□ Auditing
□ Monitoring
□ For compensation

BPMN Task 1 (Task (BPMN))

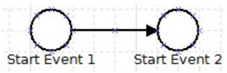
Loop details
Loop type
● Not specified
○ Standard
○ Multi-instance
Loop condition (standard):
Multi-instance loop details
□ Sequential execution

OMiLAB
www.omilab.org

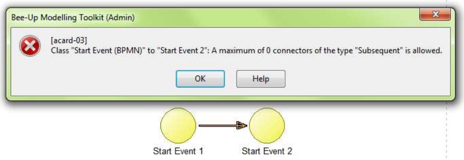
- A further distinction can be made when looking at the properties of the modelled constructs
- On the left side, using a drawing tool, we can edit the appearance of the construct, whereas
- On the right side, using a modelling tool, we can edit modelling language properties that further specify the semantics of an element, in this example of a BPMN task

Drawing- vs. Modelling Tools – Example (2)

- Drawing Tool

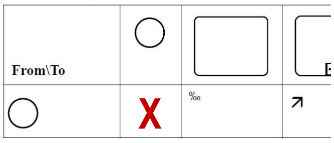


- Modelling Tool



[acard-03]
Class "Start Event (BPMN)" to "Start Event 2": A maximum of 0 connectors of the type "Subsequent" is allowed.

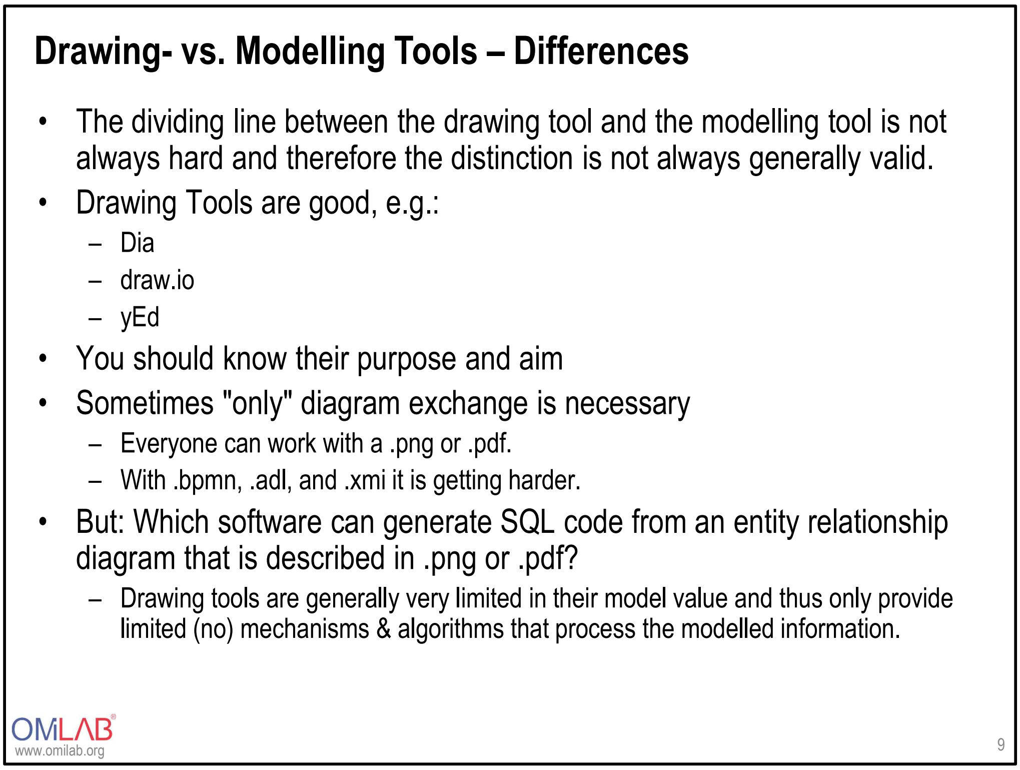
Table 7.3 – Sequence Flow Connection Rules



From\To	○	▢	▢
○	X	‰	↗

Quelle: OMG (2011) Business Process Model and Notation v2.0, www.omg.org

- Another big difference comes when thinking about the validity of the models
- In tools like PowerPoint one can naturally do whatever he/she thinks is correct, e.g., you can connect everything with everything.
- In contrast, modelling tools are aware of the grammatic rules of the modelling language syntax. As such they won’t allow invalid combinations like connecting a BPMN start event with another BPMN start event as this is prohibited by the BPMN specification (see at the bottom).

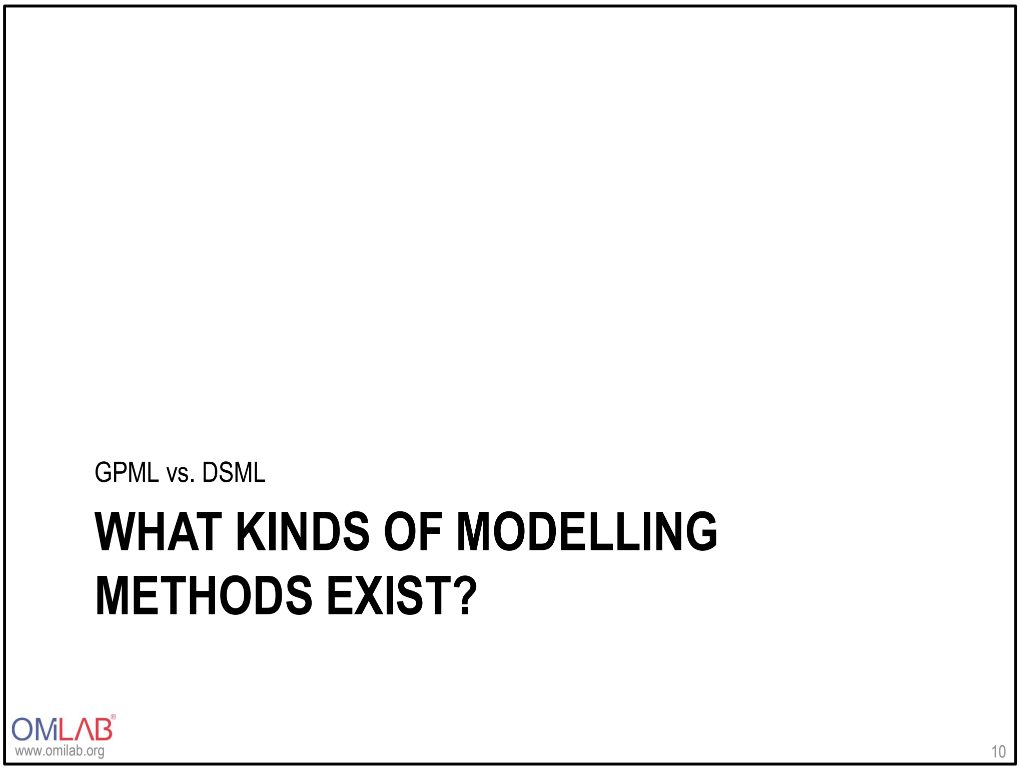


Drawing- vs. Modelling Tools – Differences

- The dividing line between the drawing tool and the modelling tool is not always hard and therefore the distinction is not always generally valid.
- Drawing Tools are good, e.g.:
  - Dia
  - draw.io
  - yEd
- You should know their purpose and aim
- Sometimes "only" diagram exchange is necessary
  - Everyone can work with a .png or .pdf.
  - With .bpmn, .adl, and .xmi it is getting harder.
- But: Which software can generate SQL code from an entity relationship diagram that is described in .png or .pdf?
  - Drawing tools are generally very limited in their model value and thus only provide limited (no) mechanisms & algorithms that process the modelled information.

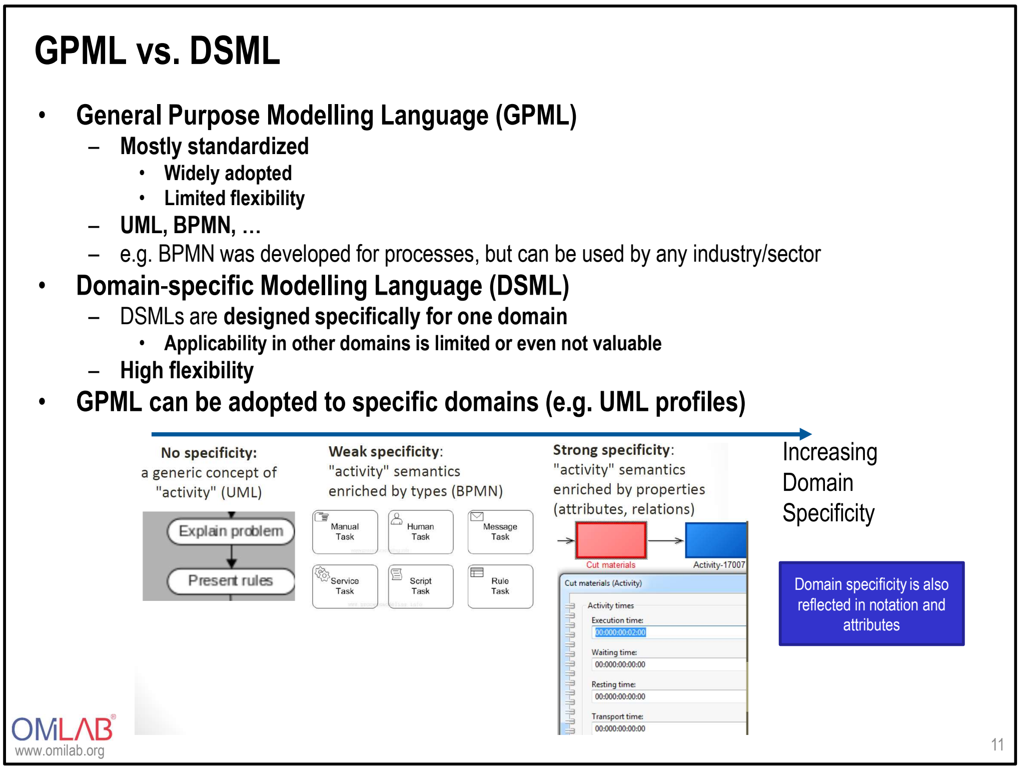
www.omilab.org

- Make sure to also give credits to the drawing tools in those aspects they are good in
- However, the different purposes and capabilities of the two categories of tools should still be emphasized!
- Especially when considering model processing by mechanisms & algorithms, one needs to use proper modelling tools!



GPML vs. DSML

WHAT KINDS OF MODELLING METHODS EXIST?



GPML vs. DSML

- **General Purpose Modelling Language (GPML)**
  - **Mostly standardized**
    - **Widely adopted**
    - **Limited flexibility**
  - **UML, BPMN, …**
  - e.g. BPMN was developed for processes, but can be used by any industry/sector
- **Domain-specific Modelling Language (DSML)**
  - DSMLs are **designed specifically for one domain**
    - **Applicability in other domains is limited or even not valuable**
  - **High flexibility**
- **GPML can be adopted to specific domains (e.g. UML profiles)**

No specificity:
a generic concept of
"activity" (UML)

Weak specificity:
"activity" semantics
enriched by types (BPMN)

Strong specificity:
"activity" semantics
enriched by properties
(attributes, relations)

Increasing
Domain
Specificity

Domain specificity is also reflected in notation and attributes

Cut materials (Activity)

Activity times

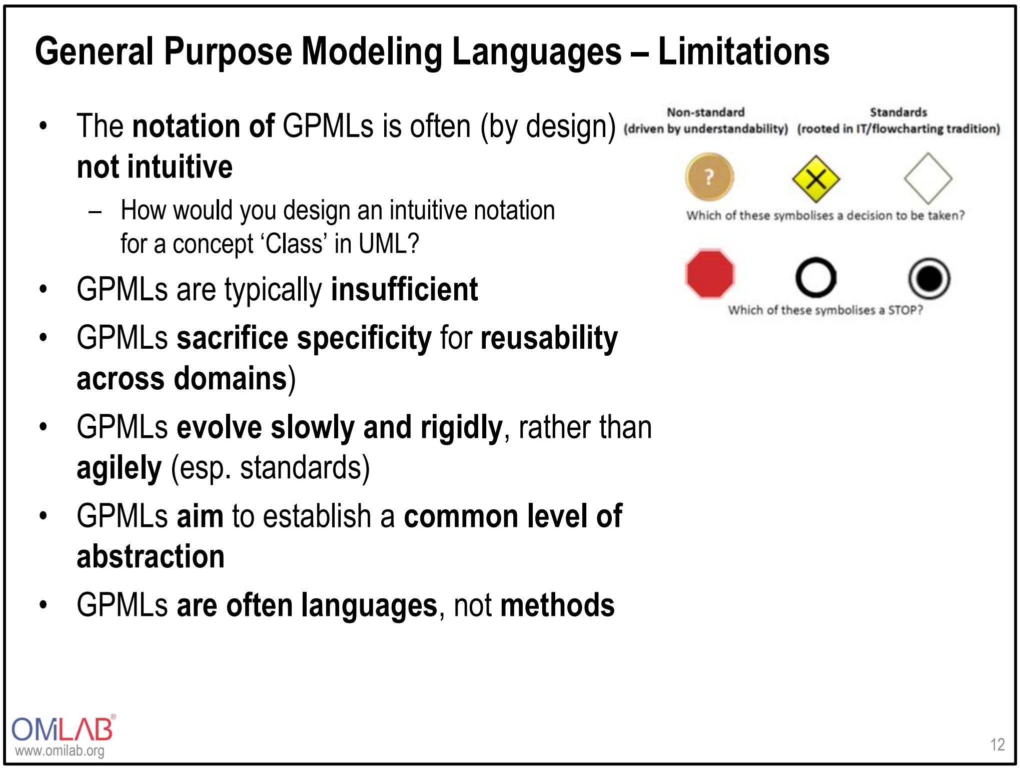
Execution time:
00:000:00:02:00

Waiting time:
00:000:00:00:00

Resting time:
00:000:00:00:00

Transport time:
00:000:00:00:00

- This slide compares general purpose modelling languages with domain-specific ones
- Make sure to emphasize that both are highly relevant for different reasons
  - GPML, mostly standardized, have wide adoption and establish industry-wide communication
  - DSML, in contrast, aim for domain-specificity in all modelling method components
    - Can help in very specific problems, esp. also related to code generation
- The figure at the bottom shows, how also GPML languages can be enriched with domain-specificity



General Purpose Modeling Languages – Limitations

- The **notation of** GPMLs is often (by design) **not intuitive**
  - How would you design an intuitive notation for a concept 'Class' in UML?
- GPMLs are typically **insufficient**
- GPMLs **sacrifice specificity** for **reusability across domains**)
- GPMLs **evolve slowly and rigidly**, rather than **agilely** (esp. standards)
- GPMLs **aim** to establish a **common level of abstraction**
- GPMLs **are often languages**, not **methods**

Non-standard (driven by understandability)

Standards (rooted in IT/flowcharting tradition)

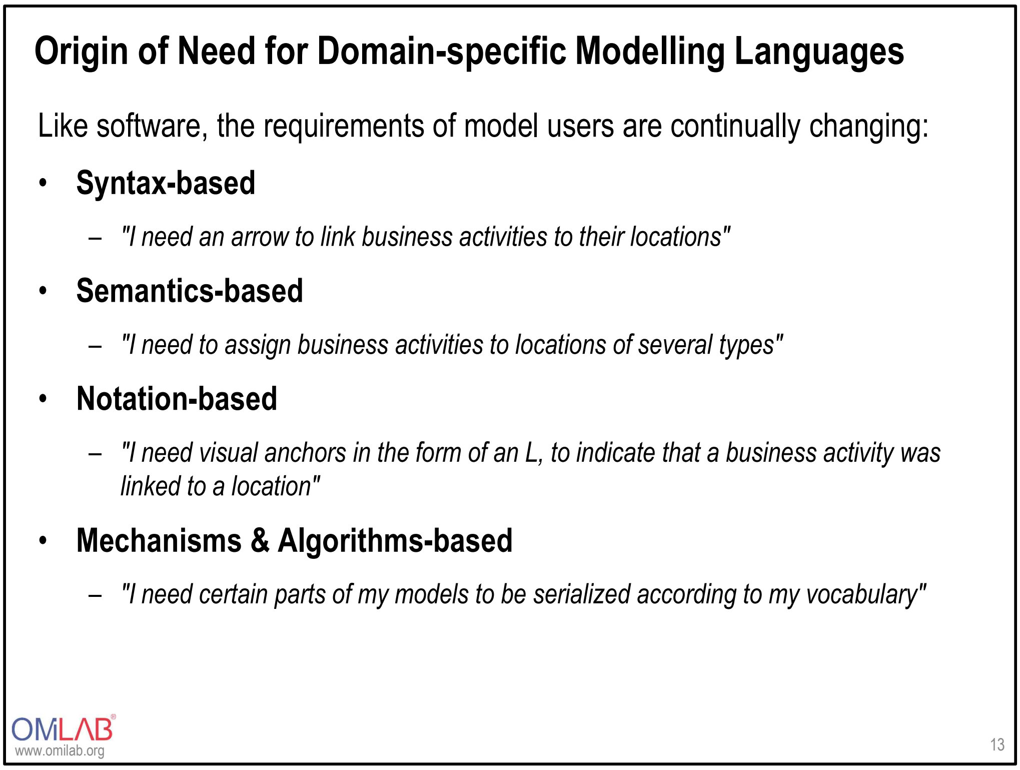
Which of these symbolises a decision to be taken?

Which of these symbolises a STOP?

OMiLAB www.omilab.org

12

- This slide shows domain-specificity in the notation and stresses, that domain-specific notations are mmostly more visually expressive (allow for intuitive interpretation)
- Exmaple question to the audience: How would you design an intuitive notation for the UML Class concept?
- The rest of the slide aims to point to other drawbacks of GPMLs

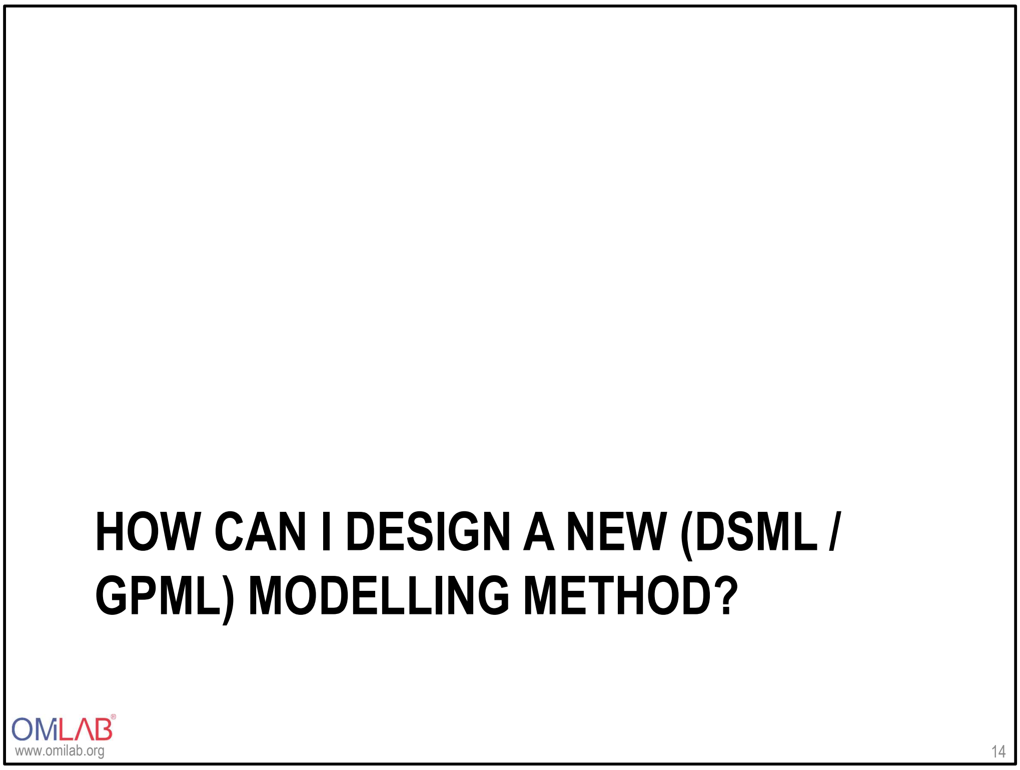


Origin of Need for Domain-specific Modelling Languages

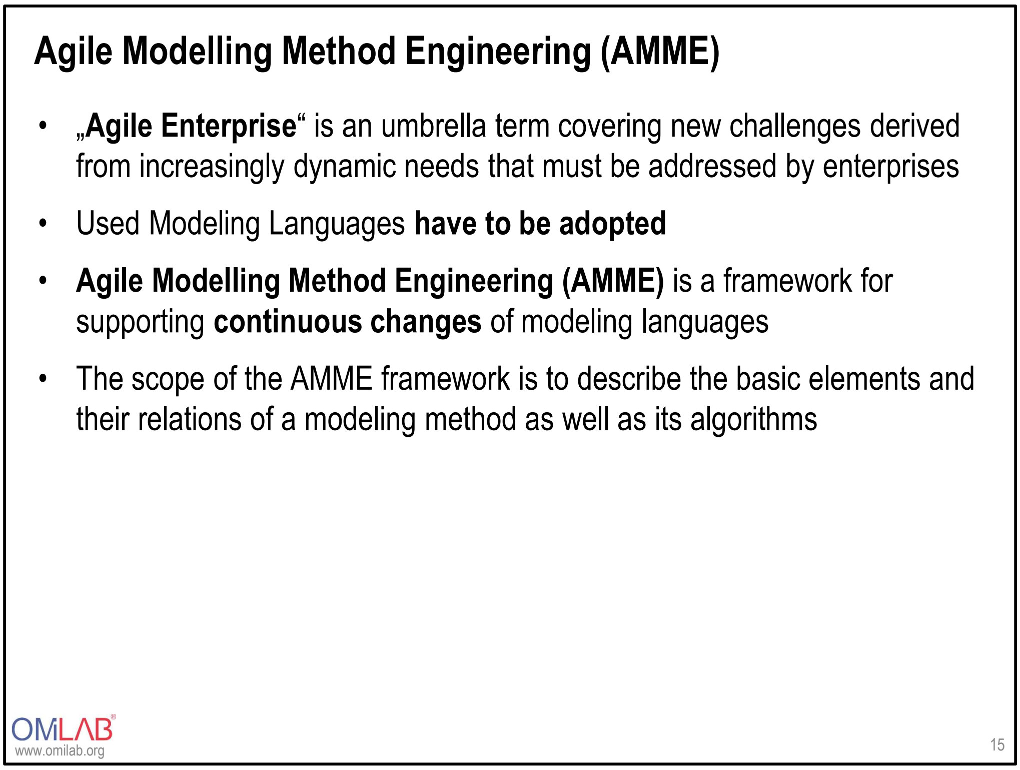
Like software, the requirements of model users are continually changing:

- **Syntax-based**
  - *"I need an arrow to link business activities to their locations"*
- **Semantics-based**
  - *"I need to assign business activities to locations of several types"*
- **Notation-based**
  - *"I need visual anchors in the form of an L, to indicate that a business activity was linked to a location"*
- **Mechanisms & Algorithms-based**
  - *"I need certain parts of my models to be serialized according to my vocabulary"*

- This slide exemplifies a few sources for domain-specific requirements
- It further shows, that domain-specificity can relate to all components of a modelling method



HOW CAN I DESIGN A NEW (DSML / GPML) MODELLING METHOD?



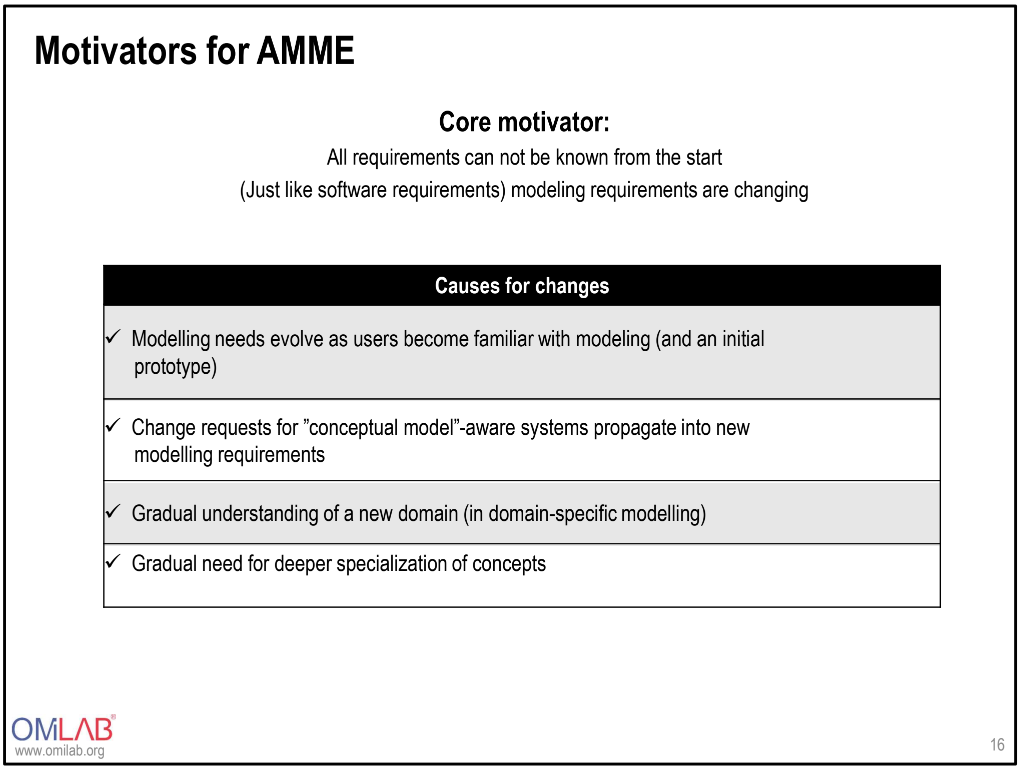
Agile Modelling Method Engineering (AMME)

- „**Agile Enterprise**“ is an umbrella term covering new challenges derived from increasingly dynamic needs that must be addressed by enterprises
- Used Modeling Languages **have to be adopted**
- **Agile Modelling Method Engineering (AMME)** is a framework for supporting **continuous changes** of modeling languages
- The scope of the AMME framework is to describe the basic elements and their relations of a modeling method as well as its algorithms

OMILAB
www.omilab.org

15

- As enterprises need to agilely adopt to changing requirements and circumstances, so do modelling methods
- AMME is a framework that borrows concepts from agile software engineering and adopts them for the conceptualization of new modelling methods



Motivators for AMME

Core motivator:

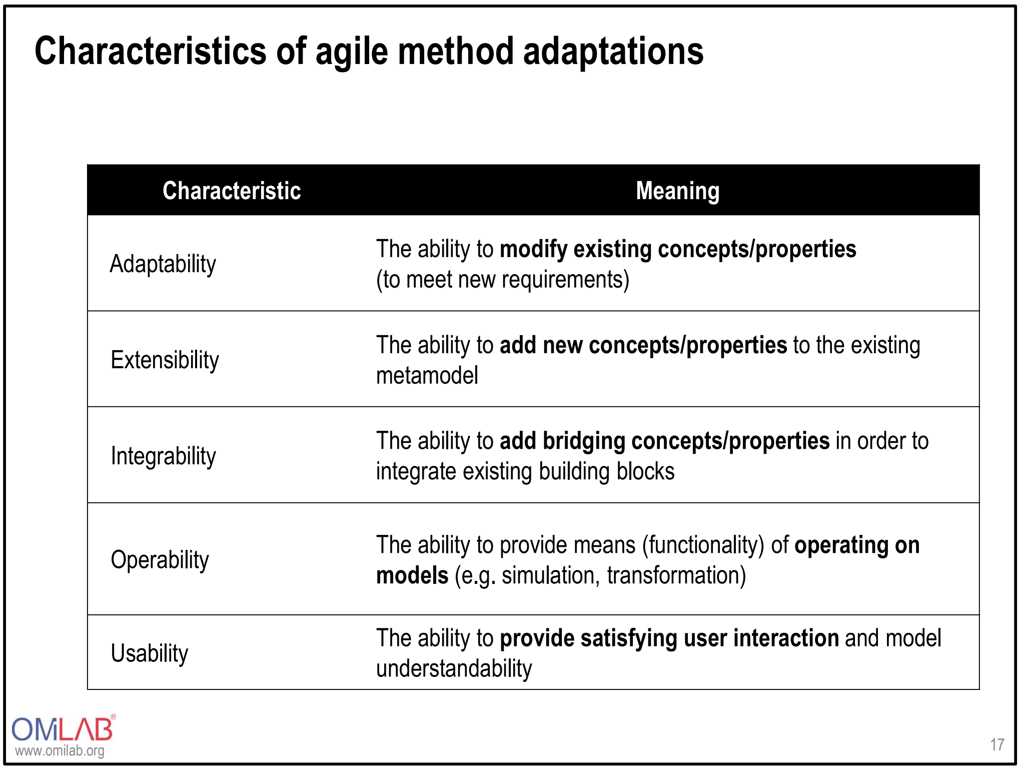
All requirements can not be known from the start

(Just like software requirements) modeling requirements are changing

Causes for changes
✓ Modelling needs evolve as users become familiar with modeling (and an initial prototype)
✓ Change requests for ”conceptual model”-aware systems propagate into new modelling requirements
✓ Gradual understanding of a new domain (in domain-specific modelling)
✓ Gradual need for deeper specialization of concepts

OMiLAB
www.omilab.org

- Describe the core motivators for adopting AMME

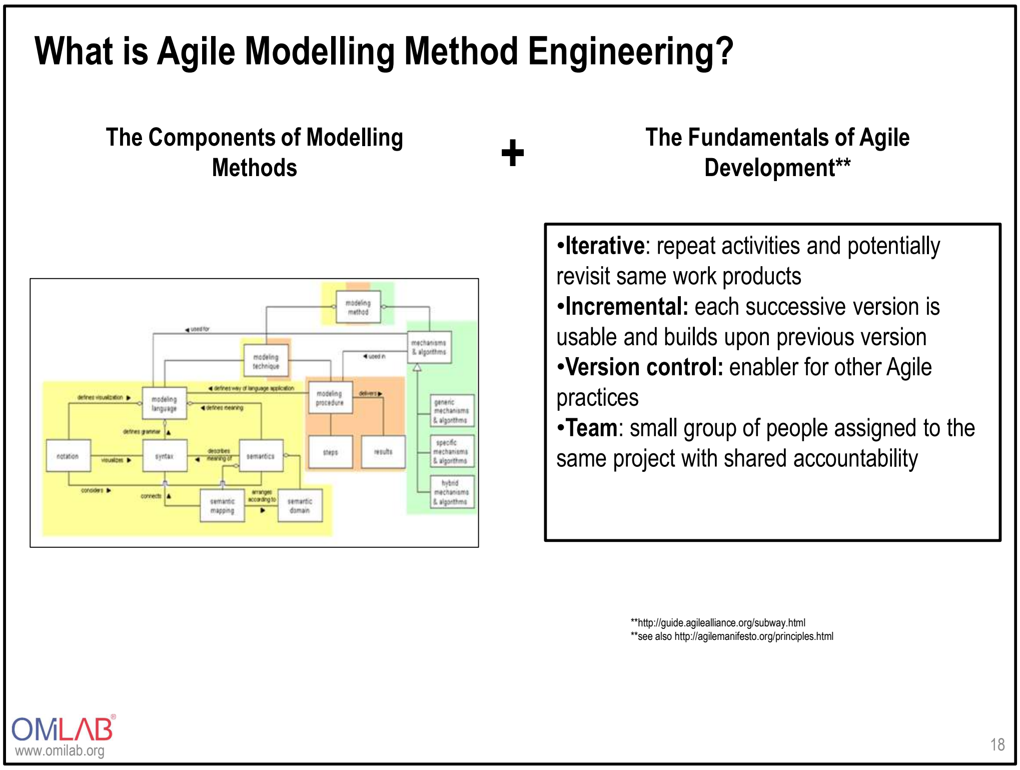


Characteristics of agile method adaptations

Characteristic	Meaning
Adaptability	The ability to **modify existing concepts/properties** (to meet new requirements)
Extensibility	The ability to **add new concepts/properties** to the existing metamodel
Integrability	The ability to **add bridging concepts/properties** in order to integrate existing building blocks
Operability	The ability to provide means (functionality) of **operating on models** (e.g. simulation, transformation)
Usability	The ability to **provide satisfying user interaction** and model understandability

OMILAB
www.omilab.org

- In response to the motivators/requirements discussed previously, AMME adheres to some specific characteristics in response
- The characteristics all relate to specific parts of the modelling method with a focus on syntactic aspects, i.e., changes of the metamodel



What is Agile Modelling Method Engineering?

The Components of Modelling Methods + **The Fundamentals of Agile Development****

modeling method
used for
mechanisms & algorithms
modeling technique
used in
defines way of language application
modeling procedure
delivers
defines visualization
modeling language
defines meaning
generic mechanisms & algorithms
defines grammar
specific mechanisms & algorithms
notation
visualizes
syntax
describes meaning of
semantics
steps
results
hybrid mechanisms & algorithms
considers
connects
semantic mapping
arranges according to
semantic domain

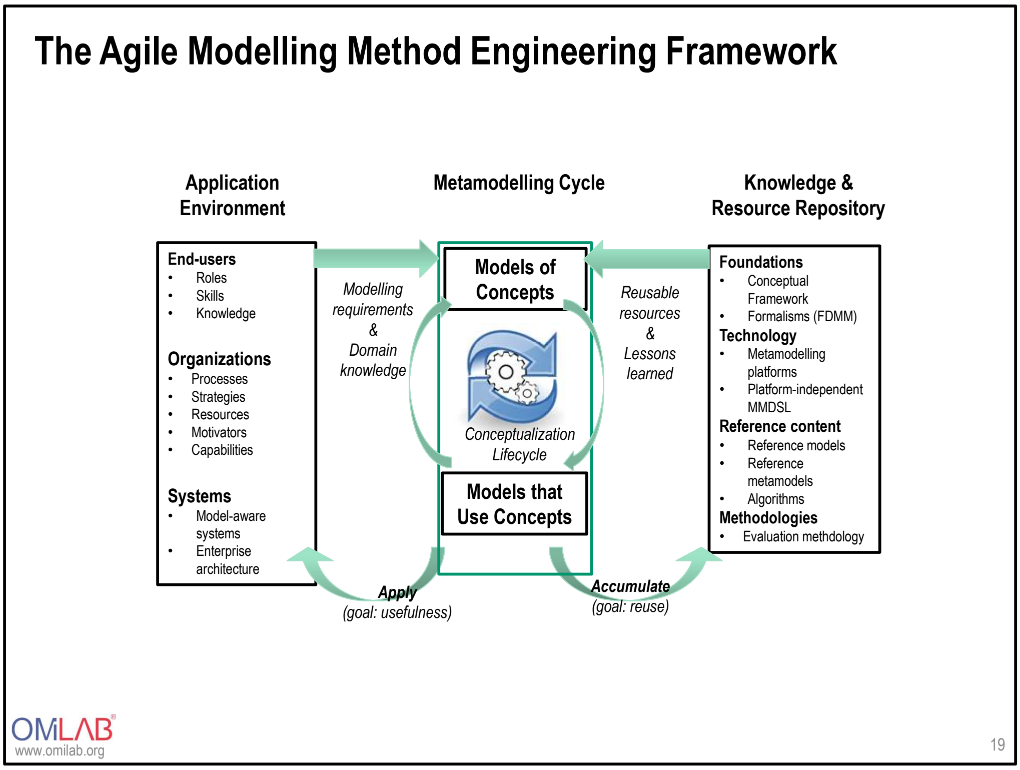
•**Iterative**: repeat activities and potentially revisit same work products
•**Incremental:** each successive version is usable and builds upon previous version
•**Version control:** enabler for other Agile practices
•**Team**: small group of people assigned to the same project with shared accountability

**http://guide.agilealliance.org/subway.html
**see also http://agilemanifesto.org/principles.html

www.omilab.org

18

- AMME is build on two pillars
    - The Components of Modeling Methods and
    - The Fundamentals of Agile Development



The Agile Modelling Method Engineering Framework

Application Environment

Metamodelling Cycle

Knowledge & Resource Repository

End-users
- Roles
- Skills
- Knowledge

Organizations
- Processes
- Strategies
- Resources
- Motivators
- Capabilities

Systems
- Model-aware systems
- Enterprise architecture

Modelling requirements & Domain knowledge

Models of Concepts

Conceptualization Lifecycle

Models that Use Concepts

Reusable resources & Lessons learned

Foundations
- Conceptual Framework
- Formalisms (FDMM)

Technology
- Metamodelling platforms
- Platform-independent MMDSL

Reference content
- Reference models
- Reference metamodels
- Algorithms

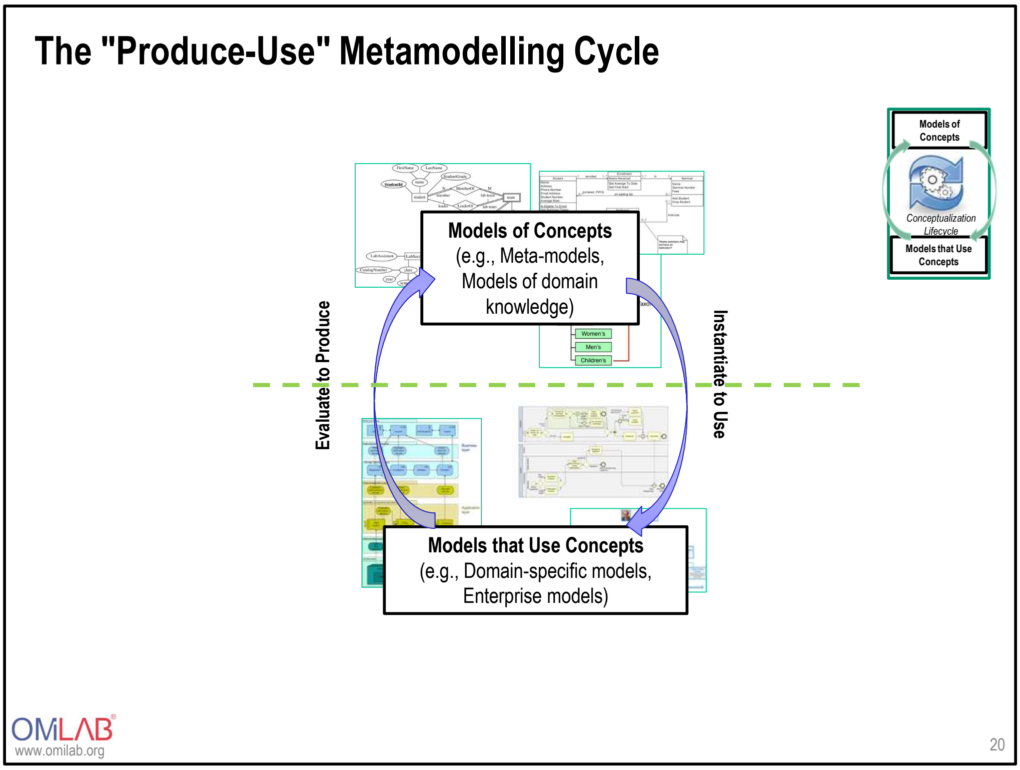
Methodologies
- Evaluation methdology

Apply (goal: usefulness)

Accumulate (goal: reuse)

OMiLAB® www.omilab.org

- This slide shows the bigger picture of AMME tat incorporates the Application Environment and the Knowledge & Resource Repository
- Requirements and domain knowledge are derived from the application environment
- Reusable resources and lessons learned from previous modelling method conceptualization projects are derived from the Knowledge & Resource Repository
- Within the Metamodeling Cycle, both inputs are combined while creating a model of concepts (i.e., a metamodel)
- This metamodel is then evaluated through instantiation, thereby creating models that use concepts (of the metamodel)
- A feedback loop closes the cycle and enables continuous improvement



The "Produce-Use" Metamodelling Cycle

Models of Concepts
(e.g., Meta-models, Models of domain knowledge)

Evaluate to Produce

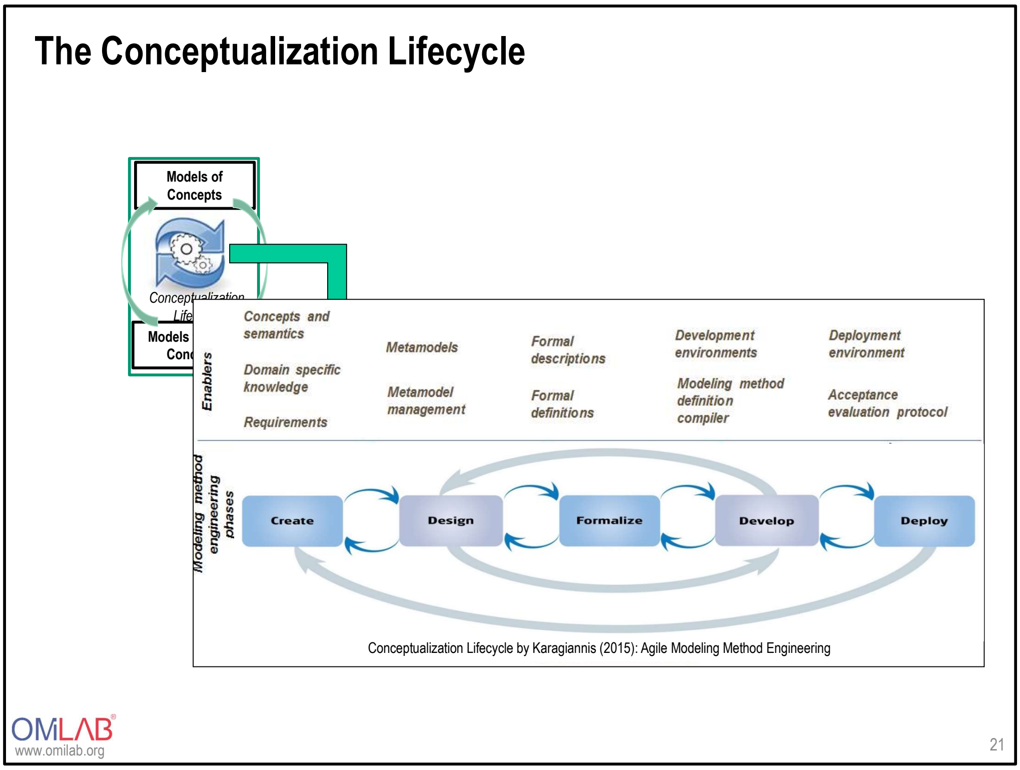
Instantiate to Use

Models that Use Concepts
(e.g., Domain-specific models, Enterprise models)

OMiLAB®
www.omilab.org

20

- This slide further explains the Produce-Use Cycle between
  - The Models of Concepts (e.g., metamodels), and
  - The models that use concepts (i.e., the concrete instantiated model)



The Conceptualization Lifecycle

Models of Concepts

Conceptualization Life

Models Conc

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

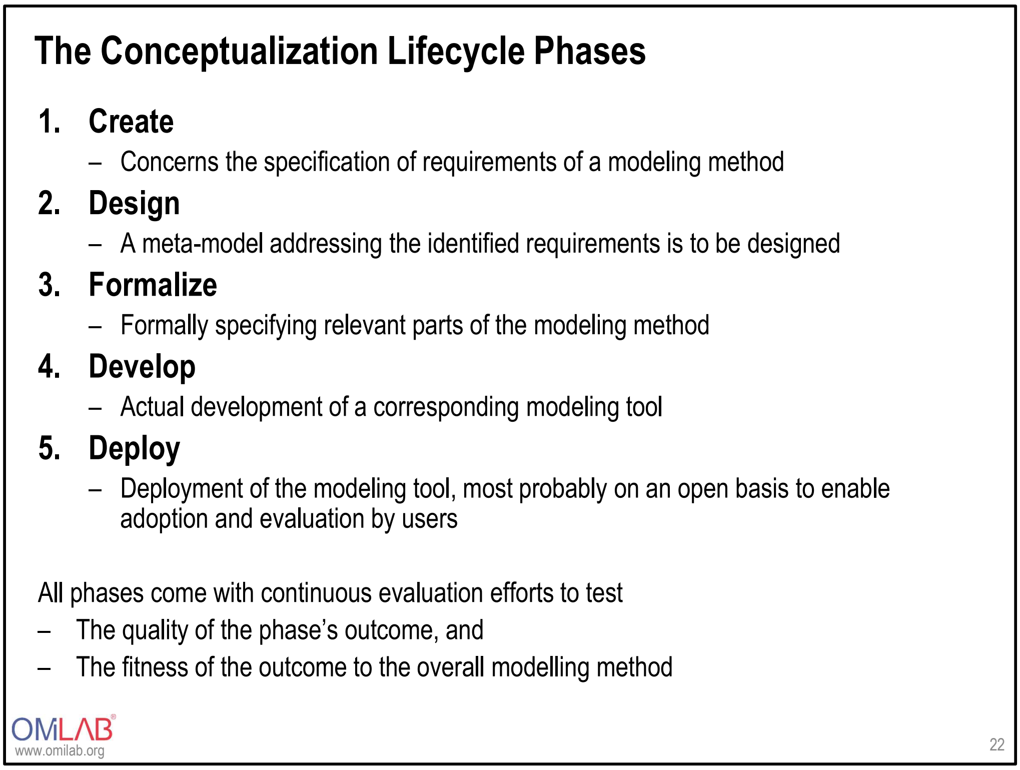
Deploy

Conceptualization Lifecycle by Karagiannis (2015): Agile Modeling Method Engineering

OMILAB
www.omilab.org

21

- Drilling once more deeper, we can see the conceptualization lifecycle in greater detail
- This slide show the different phases that form part of the AMME lifecycle
- For each phase the Enables are depicted and the relationships to other AMME phases are shown
    - The small arrows also show feedback loops within AMME



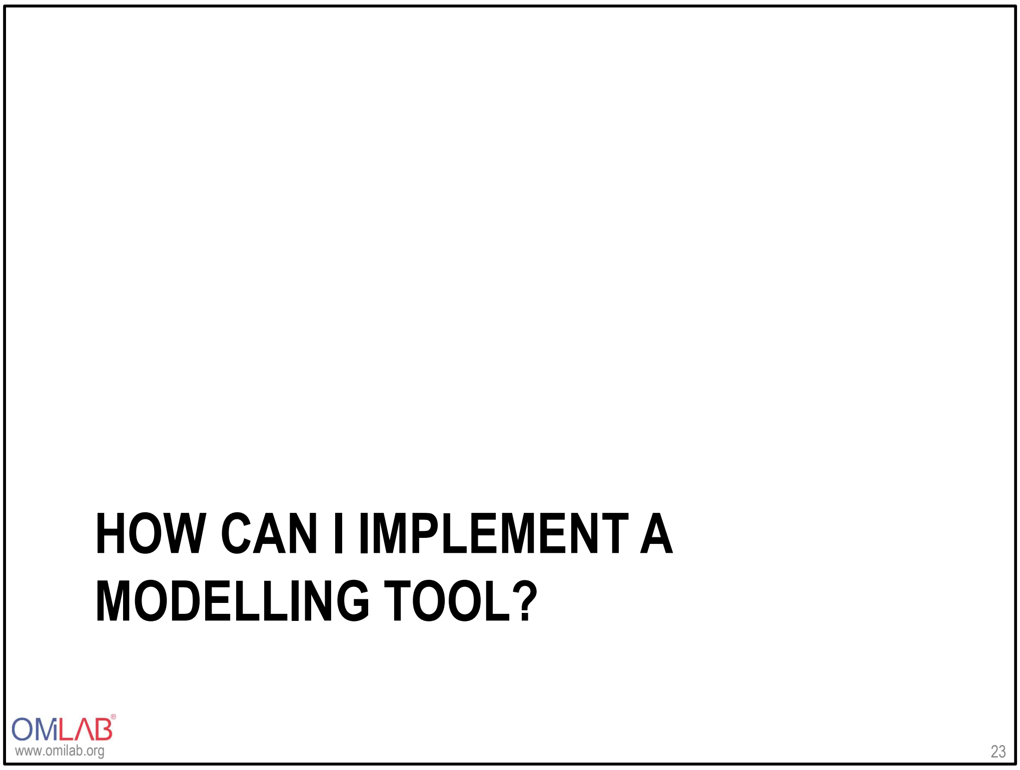
The Conceptualization Lifecycle Phases

1. **Create**
   - Concerns the specification of requirements of a modeling method
2. **Design**
   - A meta-model addressing the identified requirements is to be designed
3. **Formalize**
   - Formally specifying relevant parts of the modeling method
4. **Develop**
   - Actual development of a corresponding modeling tool
5. **Deploy**
   - Deployment of the modeling tool, most probably on an open basis to enable adoption and evaluation by users

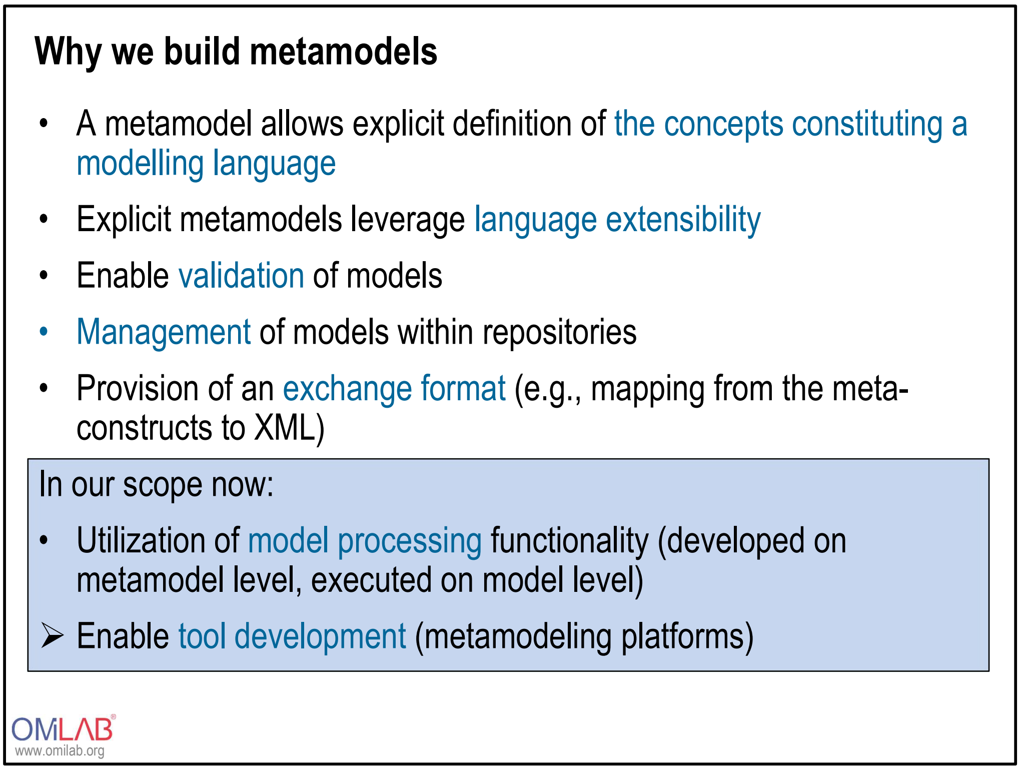
All phases come with continuous evaluation efforts to test

- The quality of the phase's outcome, and
- The fitness of the outcome to the overall modelling method

- On this slide now a more detailed description of each phase is given
- A further emphasize is given on the evaluation (the blue arrows on the previous slide) that ensure the quality of the produced artifact



HOW CAN I IMPLEMENT A MODELLING TOOL?



Why we build metamodels

- A metamodel allows explicit definition of the concepts constituting a modelling language
- Explicit metamodels leverage language extensibility
- Enable validation of models
- Management of models within repositories
- Provision of an exchange format (e.g., mapping from the meta-constructs to XML)

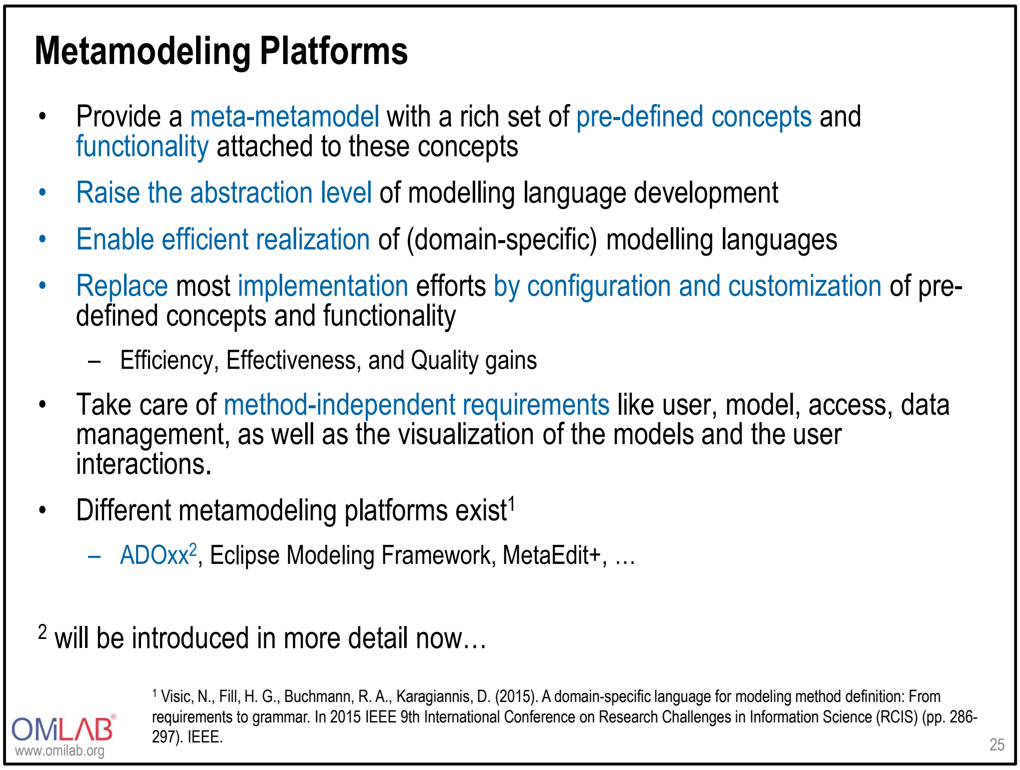
In our scope now:

- Utilization of model processing functionality (developed on metamodel level, executed on model level)

➢ Enable tool development (metamodeling platforms)

OMiLAB®
www.omilab.org

- This slide motivates why metamodel are built
- In the scope: the model processing and tool development support by metamodels which will be further detailed in the next slides



Metamodeling Platforms

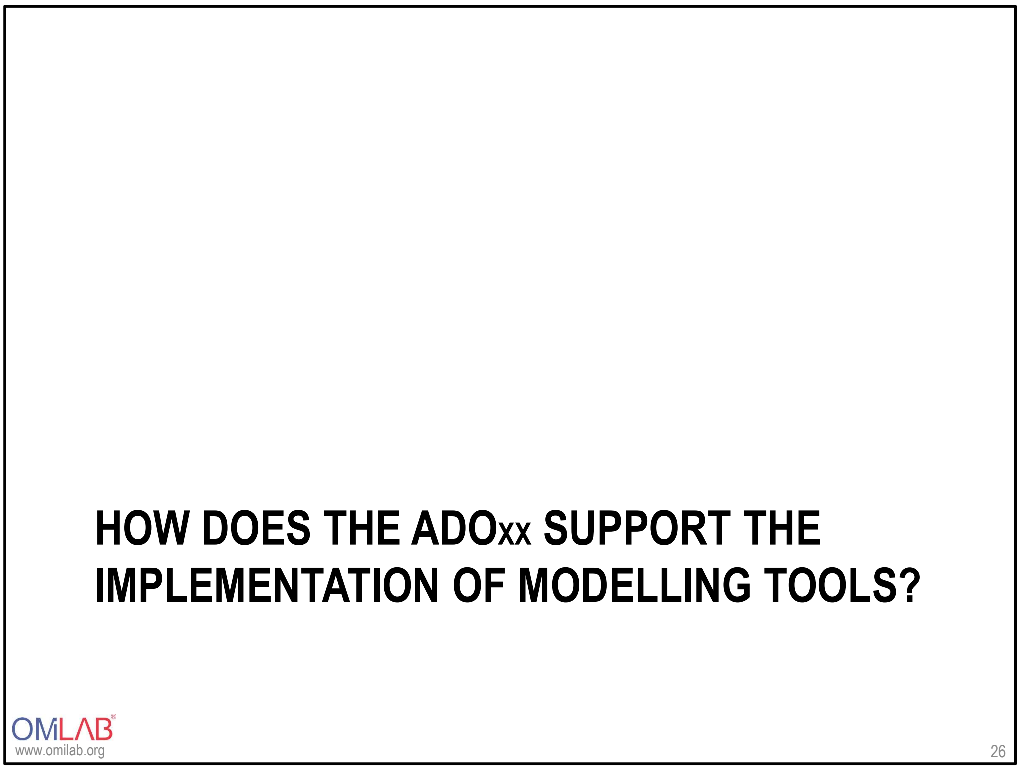
- Provide a meta-metamodel with a rich set of pre-defined concepts and functionality attached to these concepts
- Raise the abstraction level of modelling language development
- Enable efficient realization of (domain-specific) modelling languages
- Replace most implementation efforts by configuration and customization of pre-defined concepts and functionality
  - Efficiency, Effectiveness, and Quality gains
- Take care of method-independent requirements like user, model, access, data management, as well as the visualization of the models and the user interactions.
- Different metamodeling platforms exist[1]
  - ADOxx[2], Eclipse Modeling Framework, MetaEdit+, …

[2] will be introduced in more detail now…

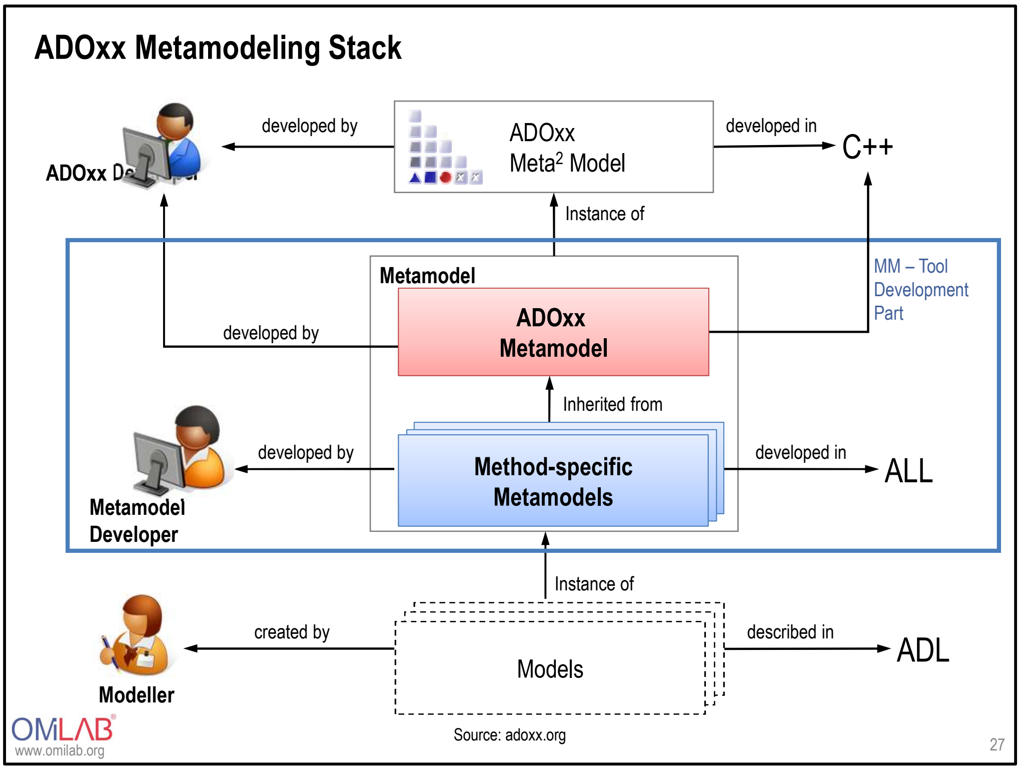
[1] Visic, N., Fill, H. G., Buchmann, R. A., Karagiannis, D. (2015). A domain-specific language for modeling method definition: From requirements to grammar. In 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS) (pp. 286-297). IEEE.

www.omilab.org

- When the realization of modellign tools is considered, one needs to think about metamodelling platforms.
- Desribe the meaning and the purpose of metamodelling platforms
- Particularly start emphasizing the efficiency in tool development that such platform provide in comparison to building modelling tools from scratch



HOW DOES THE ADOxx SUPPORT THE IMPLEMENTATION OF MODELLING TOOLS?



ADOxx Metamodeling Stack

ADOxx De er

developed by

ADOxx
Meta^2 Model

developed in

C++

Instance of

MM – Tool
Development
Part

Metamodel

ADOxx
Metamodel

developed by

Inherited from

Method-specific
Metamodels

developed by

developed in

ALL

Metamodel
Developer

Instance of

created by

Models

described in

ADL

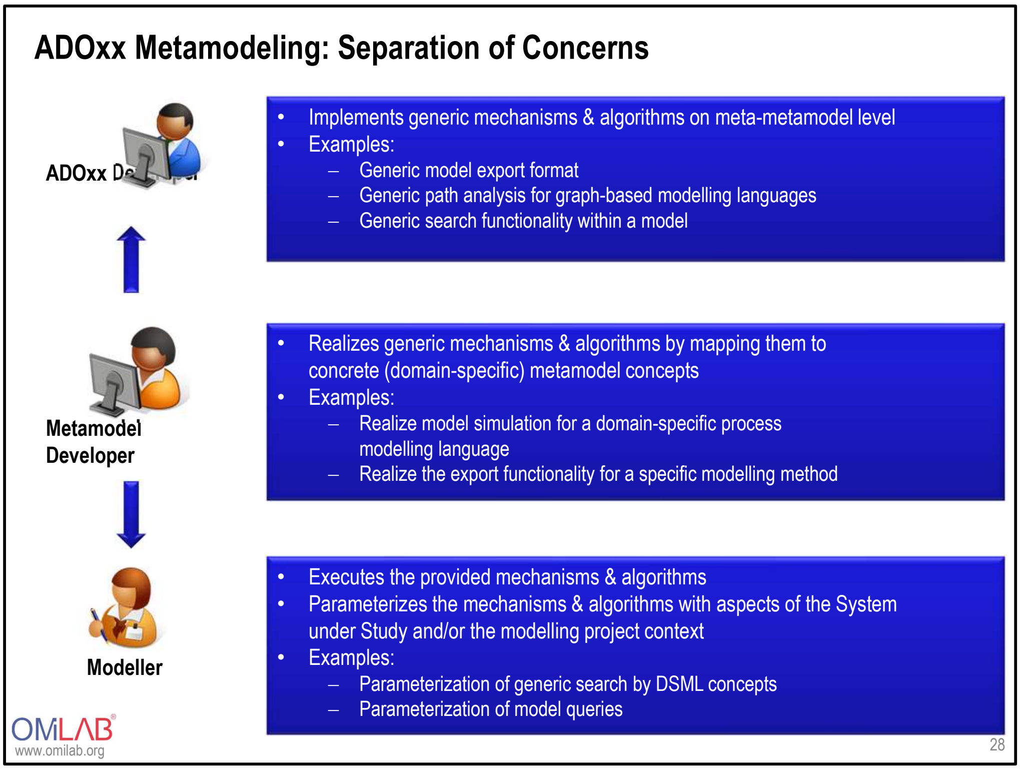
Modeller

OMILAB
www.omilab.org

Source: adoxx.org

27

- Describe the three levels and the associated roles involved in the ADOxx metamodeling stack
  - The levelled structure is though also valid for other prominent metamodeling platforms
- On top is the ADOxx developer who implements changes on the ADOxx meta-metamodel
- An instance of this metmodel is then provided to the metamodel developer as ADOxx Metamodel
- The metamodel developer then introduces his/her method-specific metamodel by inheriting from the ADOxx metamodel concepts
- Eventually, the modeler instantiates the method-specific metamodel while creating models.
- This slide also shows the different quantities on the different meta-levels, i.e.,
  - 1 ADOxx meta-metamodel
  - 1 ADOxx metamodel
  - 1..* method-specific metamodel – all inheriting from the 1 ADOxx metamodel
  - 1..* created models, each of which instanciated from one particular metamodel



ADOxx Metamodeling: Separation of Concerns

ADOxx Developer

- Implements generic mechanisms & algorithms on meta-metamodel level
- Examples:
  - Generic model export format
  - Generic path analysis for graph-based modelling languages
  - Generic search functionality within a model

Metamodel Developer

- Realizes generic mechanisms & algorithms by mapping them to concrete (domain-specific) metamodel concepts
- Examples:
  - Realize model simulation for a domain-specific process modelling language
  - Realize the export functionality for a specific modelling method

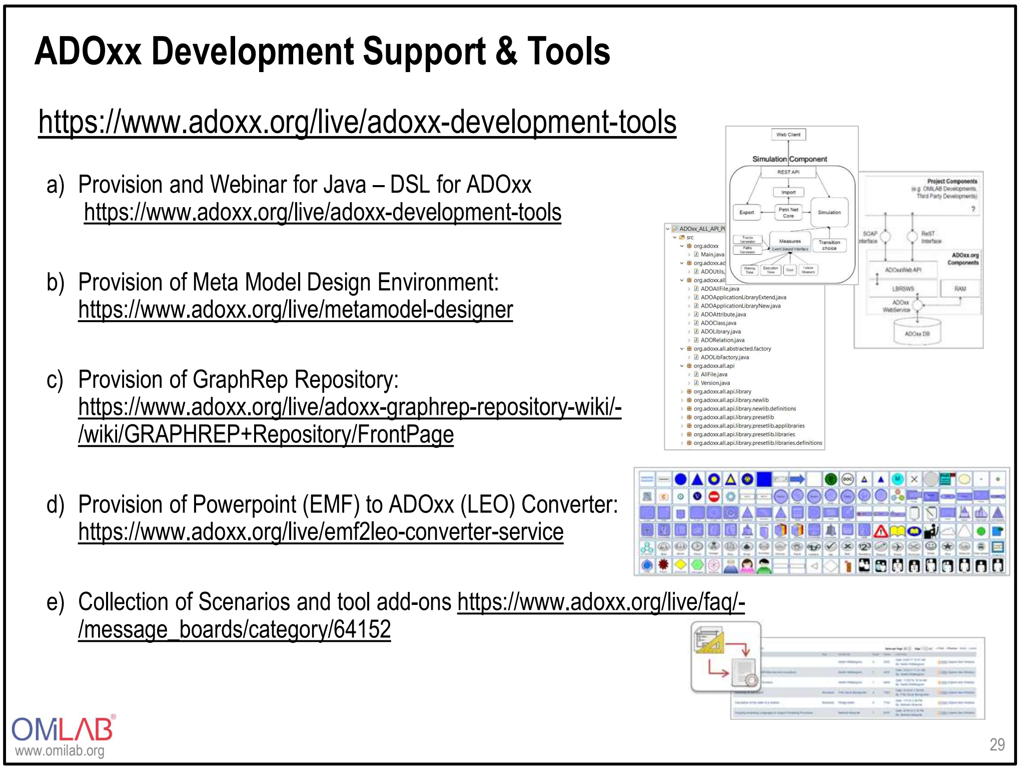
Modeller

- Executes the provided mechanisms & algorithms
- Parameterizes the mechanisms & algorithms with aspects of the System under Study and/or the modelling project context
- Examples:
  - Parameterization of generic search by DSML concepts
  - Parameterization of model queries

OMiLAB®
www.omilab.org

28

- This slide further shows the separation of concerns employed by ADOxx
- It should further emphasize that lots of the implementation efforts are taken care of by the ADOxx developer
- What is left for metamodel developer and modeller is mostly customization and parameterization of pre-defined functionality and algorithms
- Emphasize the mitigating role of the metamodel developer who
  - Translates modeller requirements into metamodel design decisions, and
  - Provides metamodel development feedback and requests for meta-metamodel functionality to the ADOxx developer



ADOxx Development Support & Tools

https://www.adoxx.org/live/adoxx-development-tools

a) Provision and Webinar for Java – DSL for ADOxx
https://www.adoxx.org/live/adoxx-development-tools

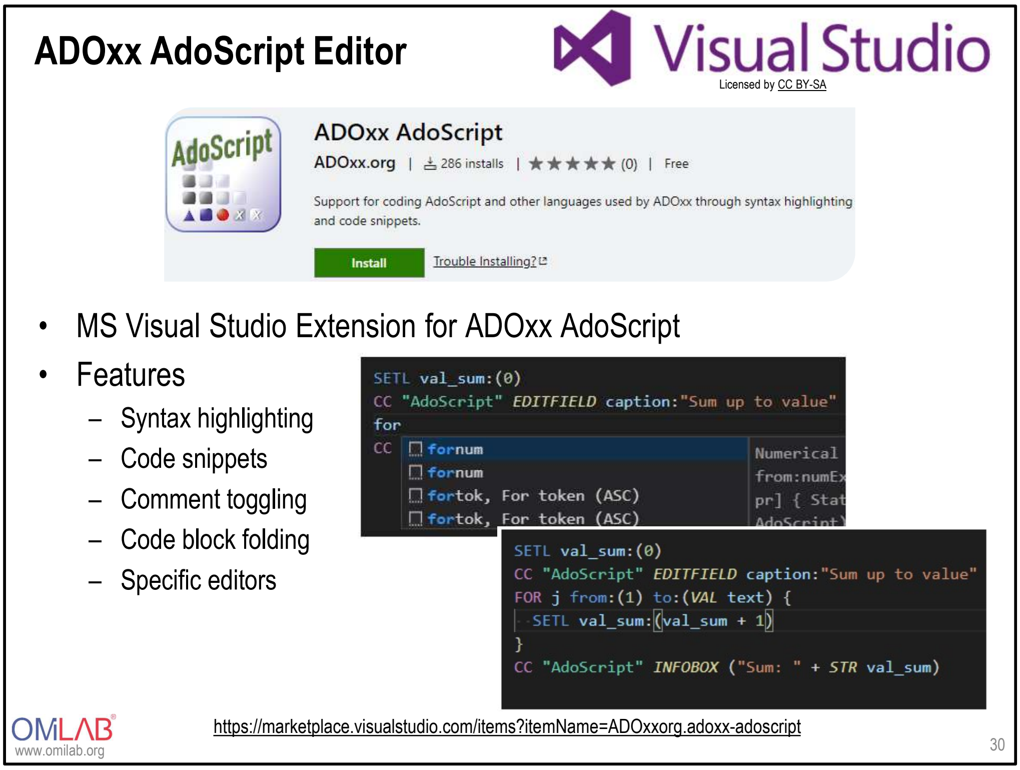
b) Provision of Meta Model Design Environment:
https://www.adoxx.org/live/metamodel-designer

c) Provision of GraphRep Repository:
https://www.adoxx.org/live/adoxx-graphrep-repository-wiki/-/wiki/GRAPHREP+Repository/FrontPage

d) Provision of Powerpoint (EMF) to ADOxx (LEO) Converter:
https://www.adoxx.org/live/emf2leo-converter-service

e) Collection of Scenarios and tool add-ons https://www.adoxx.org/live/faq/-/message_boards/category/64152

- The ADOxx community also provides a rich set of further development support and development tools
- Pick one or two to explain in greater detail



ADOxx AdoScript Editor

ADOxx AdoScript

ADOxx.org | 286 installs | ★★★★★ (0) | Free

Support for coding AdoScript and other languages used by ADOxx through syntax highlighting and code snippets.

- MS Visual Studio Extension for ADOxx AdoScript
- Features
  - Syntax highlighting
  - Code snippets
  - Comment toggling
  - Code block folding
  - Specific editors

```
SETL val_sum:(0)
CC "AdoScript" EDITFIELD caption:"Sum up to value"
for
CC  □ fornum                       Numerical
    □ fornum                       from:numEx
    □ fortok, For token (ASC)      pr] { Stat
    □ fortok, For token (ASC)      AdoScript
```

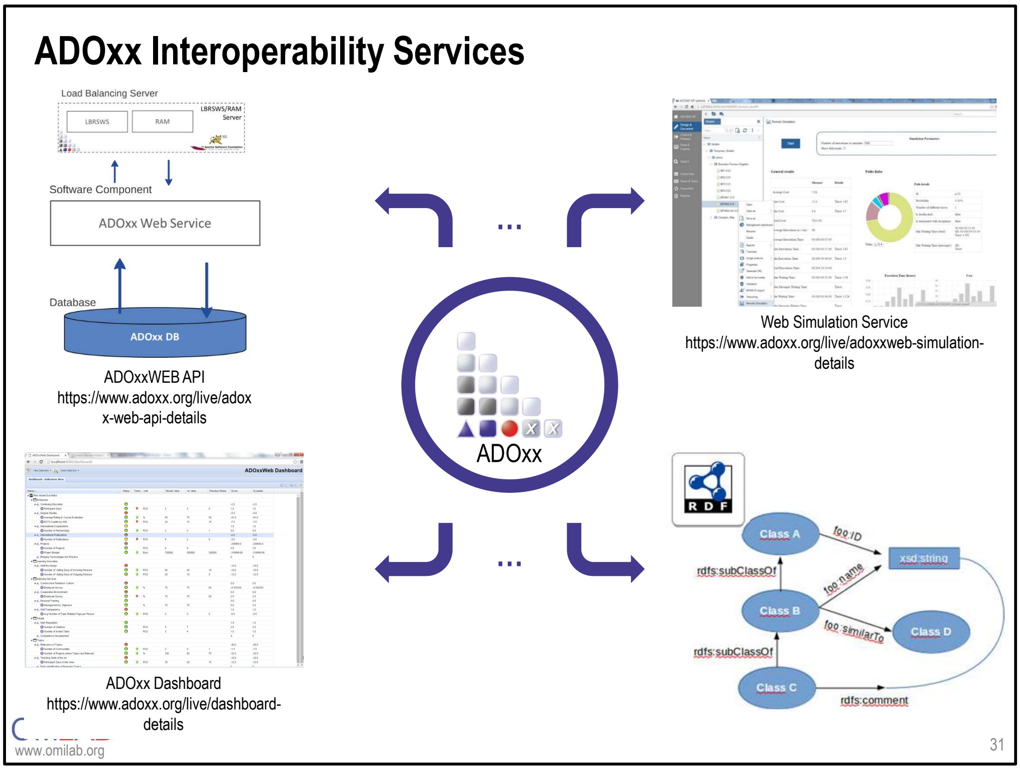
```
SETL val_sum:(0)
CC "AdoScript" EDITFIELD caption:"Sum up to value"
FOR j from:(1) to:(VAL text) {
  SETL val_sum:(val_sum + 1)
}
CC "AdoScript" INFOBOX ("Sum: " + STR val_sum)
```

OMiLAB
www.omilab.org

https://marketplace.visualstudio.com/items?itemName=ADOxxorg.adoxx-adoscript

30

- When aiming to apply metamodeling, one also needs to consider the realization of mechanisms & algorithms
- In the ADOxx world, M&A are realized wich AdoScript
- This slide introduced the ADOxx AdoScript editor which is available freely as an MS Visual Studio plugin and supports efficient development of AdoScript code



ADOxx Interoperability Services

ADOxxWEB API
https://www.adoxx.org/live/adoxx-web-api-details

Web Simulation Service
https://www.adoxx.org/live/adoxxweb-simulation-details

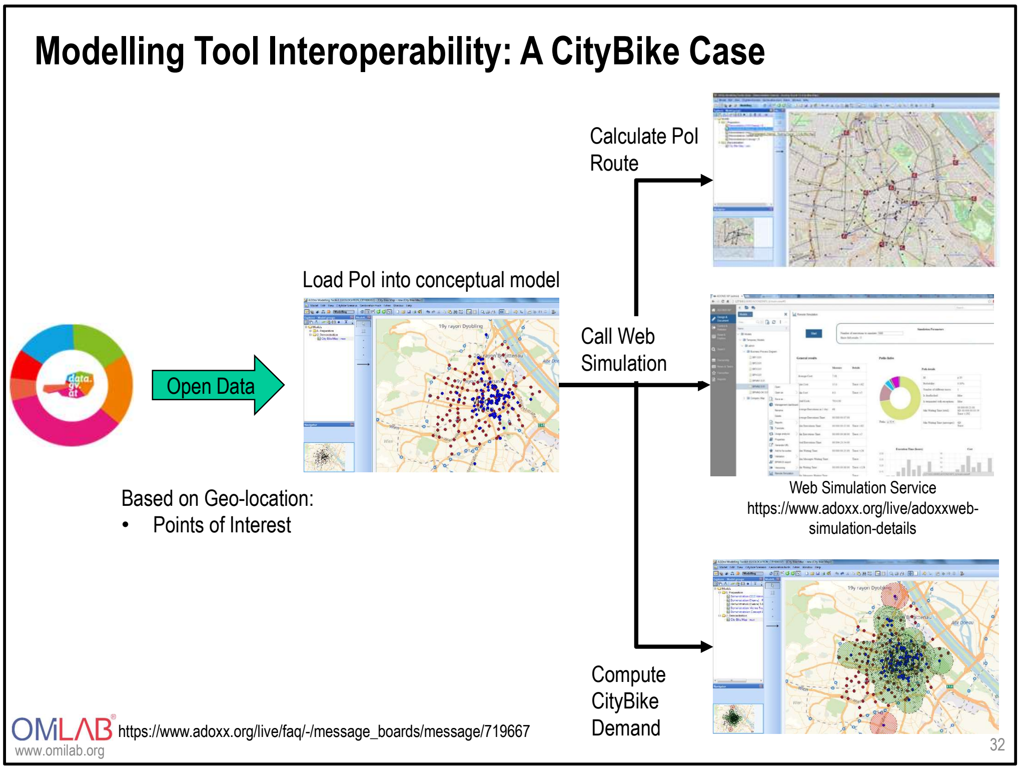
ADOxx Dashboard
https://www.adoxx.org/live/dashboard-details

www.omilab.org

31

- This slide explains four core services that enable interoperability for ADOxx based modelling tools
- The ADOxxWEB API (top left) enables the invocation of WebServices within ADOxx and also to connect to a running ADOxx instance through a HTTP Requests
- The Web Simulation service (top right) enables a lightweight and handy integration of a web dashboard to visualize ADOxx simulation results in an appealing way
- The ADOxx Dashboard service (bottom left) similarly enables reports derived from analyzing ADOxx models in an appealing way
- The RDF Transformation service (bottom right) enables the efficient serialization of ADOxx model contents in RDF format
- Emphasize that there exist much more, indicated by the three dots (…)

Picture licensed by CC BY-SA



Modelling Tool Interoperability: A CityBike Case

Open Data

Load PoI into conceptual model

Based on Geo-location:
- Points of Interest

Calculate PoI Route

Call Web Simulation

Web Simulation Service
https://www.adoxx.org/live/adoxxweb-simulation-details

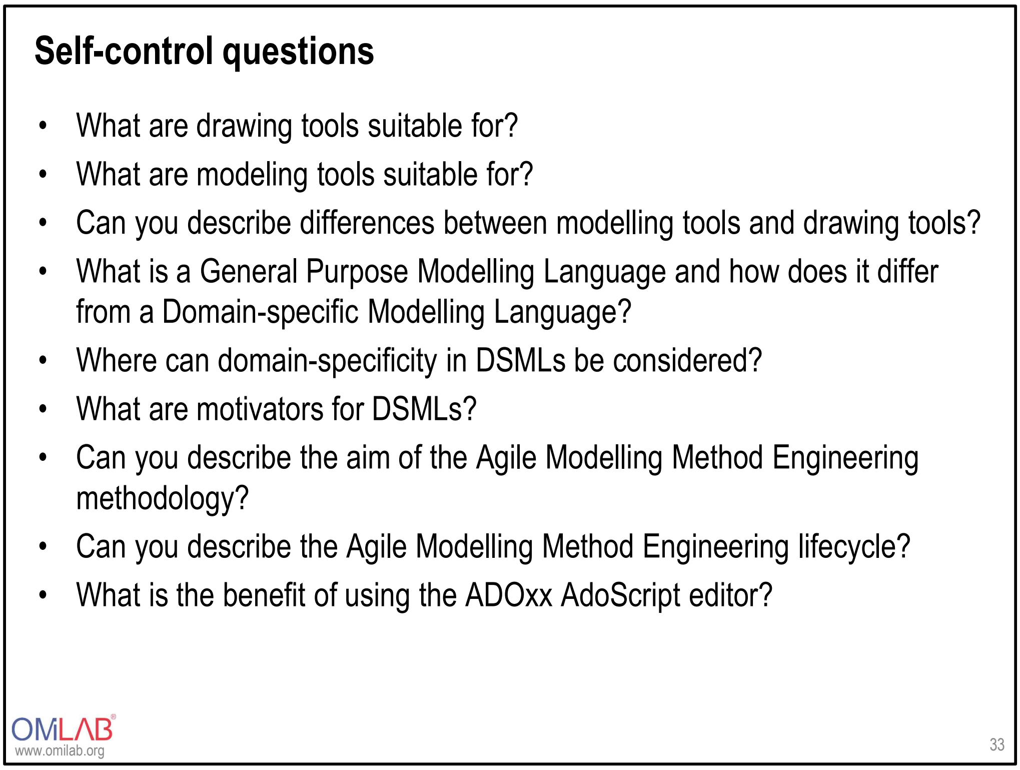
Compute CityBike Demand

OMiLAB® https://www.adoxx.org/live/faq/-/message_boards/message/719667

www.omilab.org

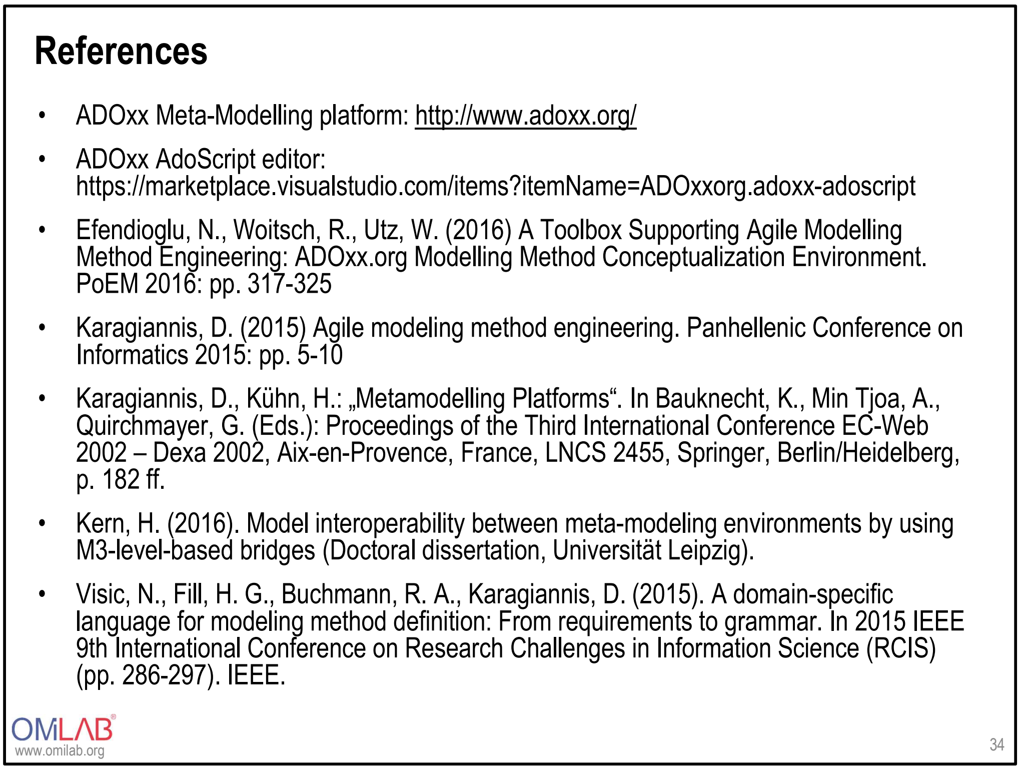
32

- This slide shows one example of interoperability based on the CityBike case
- The case uses openly available governmental data and loads it into the modelling environment
- It then processes the information by executing internal services (route planning, bike demand planning) and integrating external services like the web simulation



Self-control questions

- What are drawing tools suitable for?
- What are modeling tools suitable for?
- Can you describe differences between modelling tools and drawing tools?
- What is a General Purpose Modelling Language and how does it differ from a Domain-specific Modelling Language?
- Where can domain-specificity in DSMLs be considered?
- What are motivators for DSMLs?
- Can you describe the aim of the Agile Modelling Method Engineering methodology?
- Can you describe the Agile Modelling Method Engineering lifecycle?
- What is the benefit of using the ADOxx AdoScript editor?



References

- ADOxx Meta-Modelling platform: http://www.adoxx.org/
- ADOxx AdoScript editor: https://marketplace.visualstudio.com/items?itemName=ADOxxorg.adoxx-adoscript
- Efendioglu, N., Woitsch, R., Utz, W. (2016) A Toolbox Supporting Agile Modelling Method Engineering: ADOxx.org Modelling Method Conceptualization Environment. PoEM 2016: pp. 317-325
- Karagiannis, D. (2015) Agile modeling method engineering. Panhellenic Conference on Informatics 2015: pp. 5-10
- Karagiannis, D., Kühn, H.: „Metamodelling Platforms“. In Bauknecht, K., Min Tjoa, A., Quirchmayer, G. (Eds.): Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002, Aix-en-Provence, France, LNCS 2455, Springer, Berlin/Heidelberg, p. 182 ff.
- Kern, H. (2016). Model interoperability between meta-modeling environments by using M3-level-based bridges (Doctoral dissertation, Universität Leipzig).
- Visic, N., Fill, H. G., Buchmann, R. A., Karagiannis, D. (2015). A domain-specific language for modeling method definition: From requirements to grammar. In 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS) (pp. 286-297). IEEE.