Model Quality and Quantity

Parastoo Mohagheghi

SINTEF

Norwegian University of Science and Technology

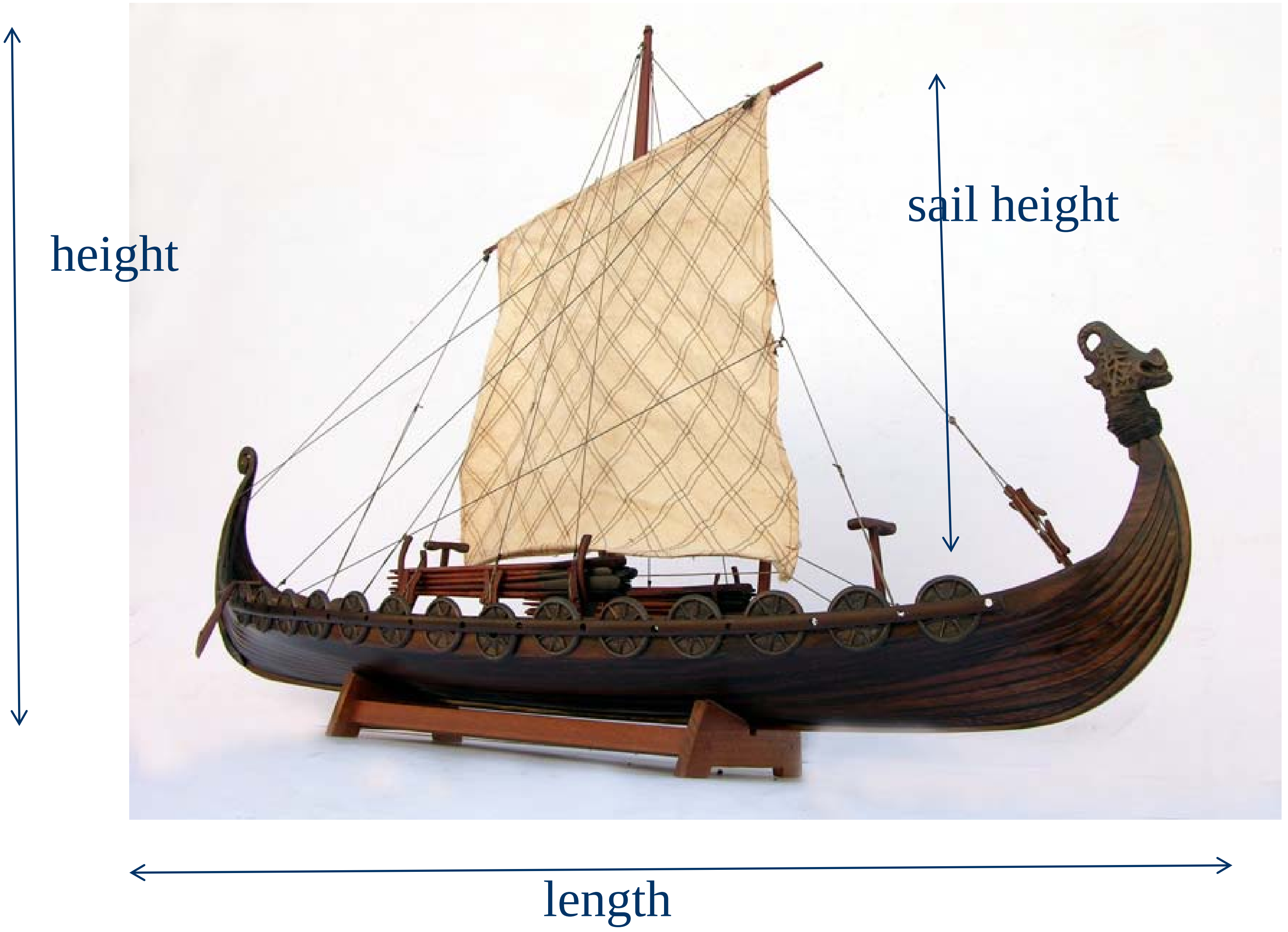
Norway

Outline

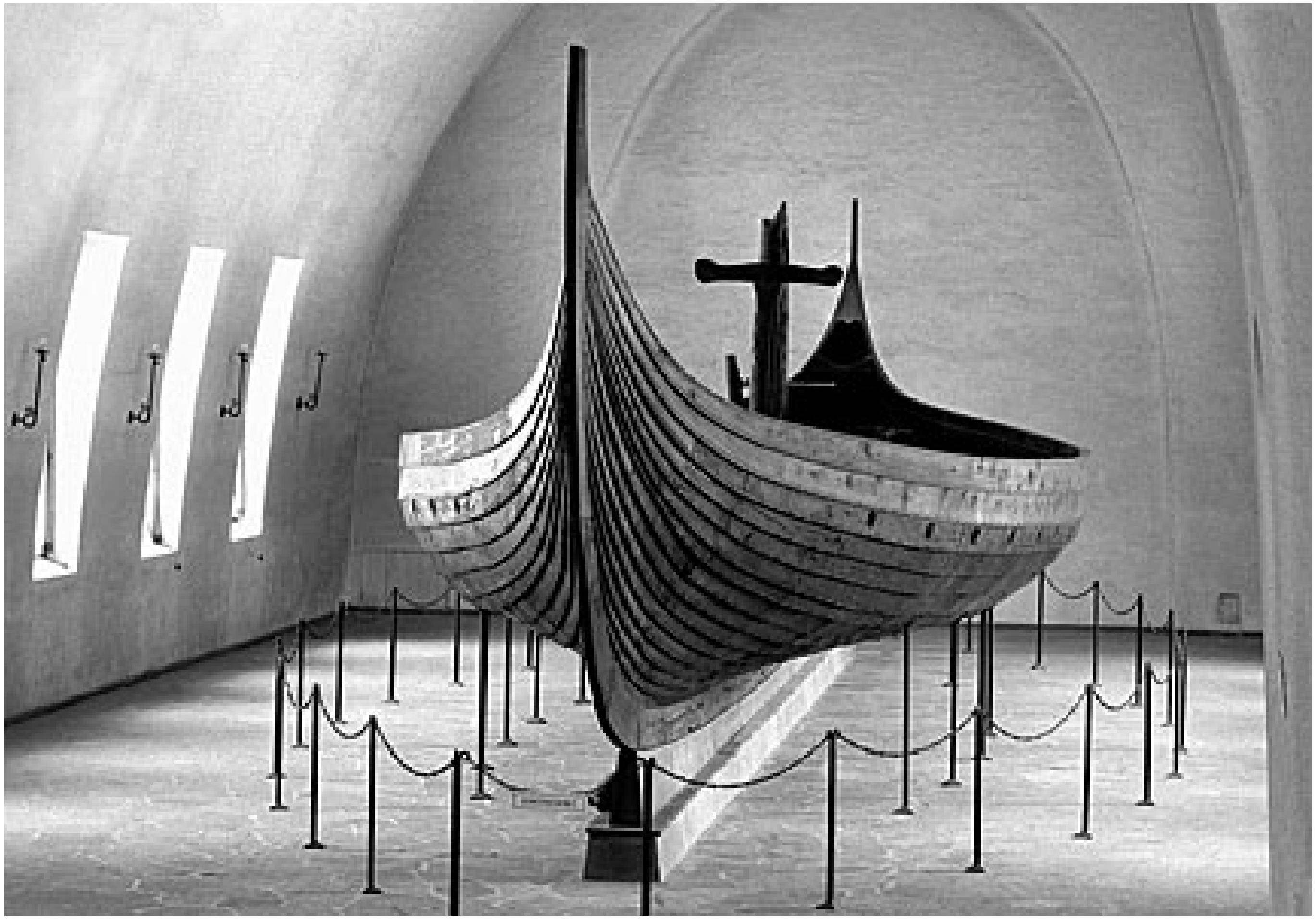
- What a model is
- What model quality means
- How to improve it
- How to measure it
- Conclusions

A few words on WHAT A MODEL IS in this context

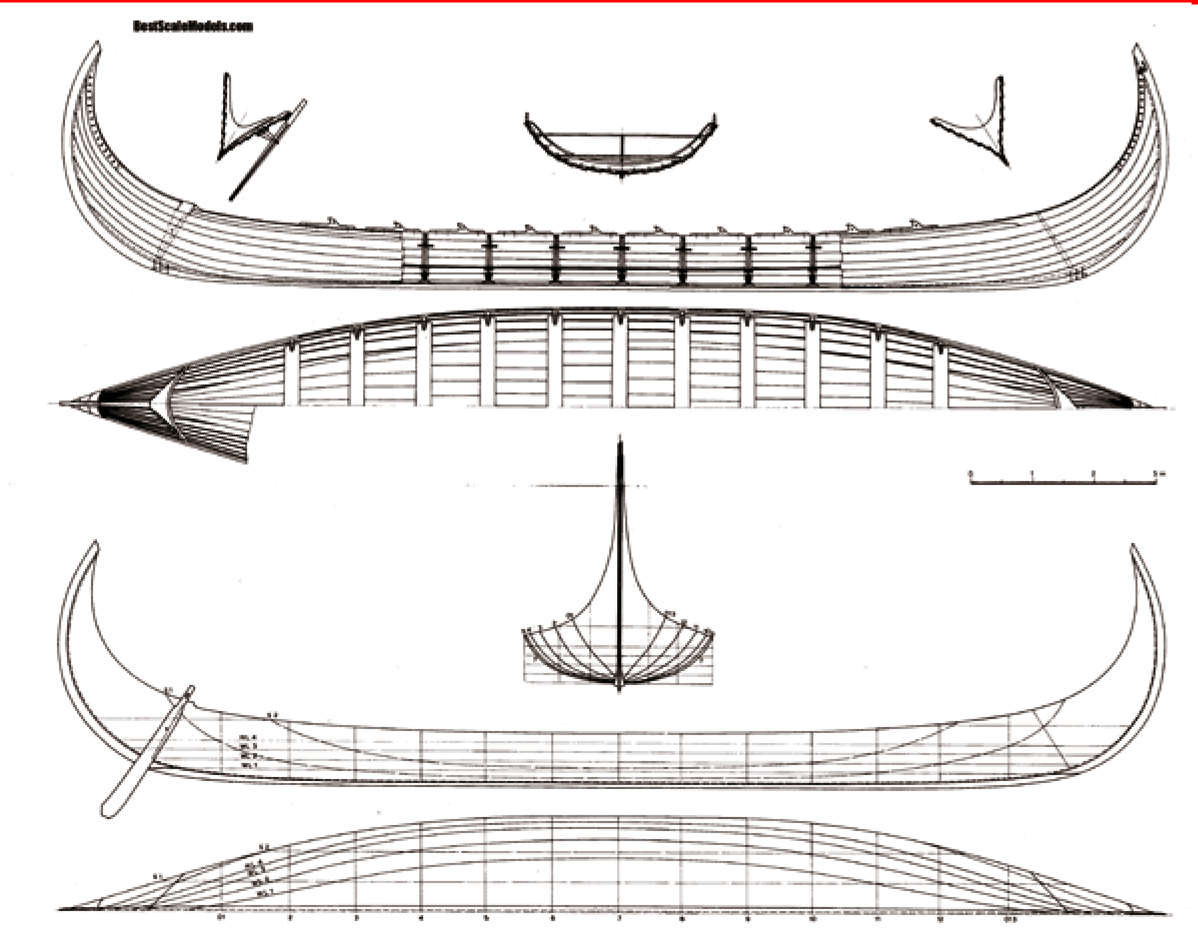
Model of a Viking Ship



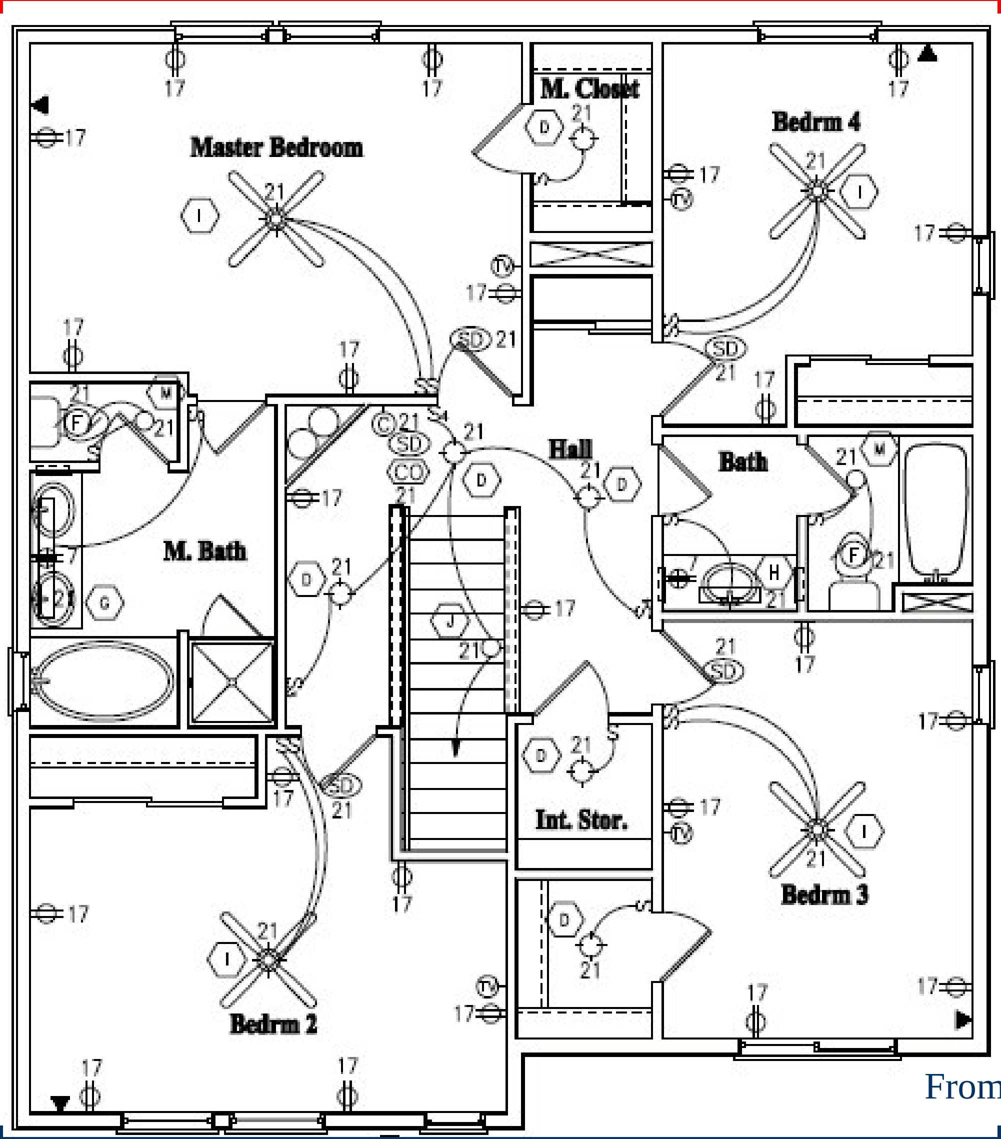
A real example: Gokstadskipet



Sketch of a Viking ship



Blueprint of a house

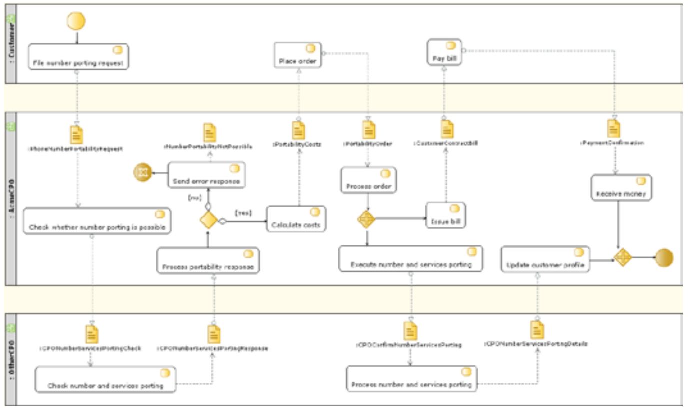


From constructionjargon.com

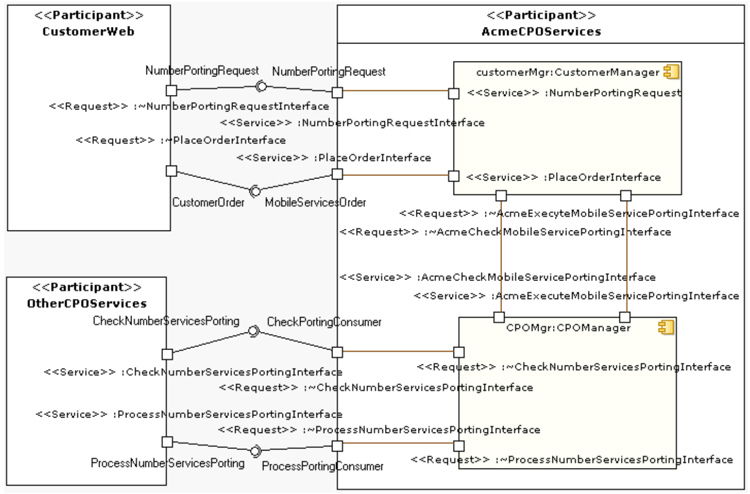
Simulations

Models in the software world

Business process model

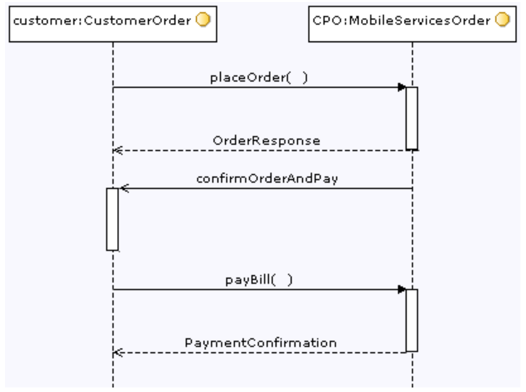


Component model



<<Participant>> CustomerWeb
<<Participant>> AcmeCPOServices
<<Participant>> OtherCPOServices
customerMgr:CustomerManager
CPOMgr:CPOManager
NumberPortingRequest
CustomerOrder
MobileServicesOrder
CheckNumberServicesPorting
CheckPortingConsumer
ProcessNumberServicesPorting
ProcessPortingConsumer

Sequence diagram



interfaces
source code
test cases
executable code

Why do we build models?

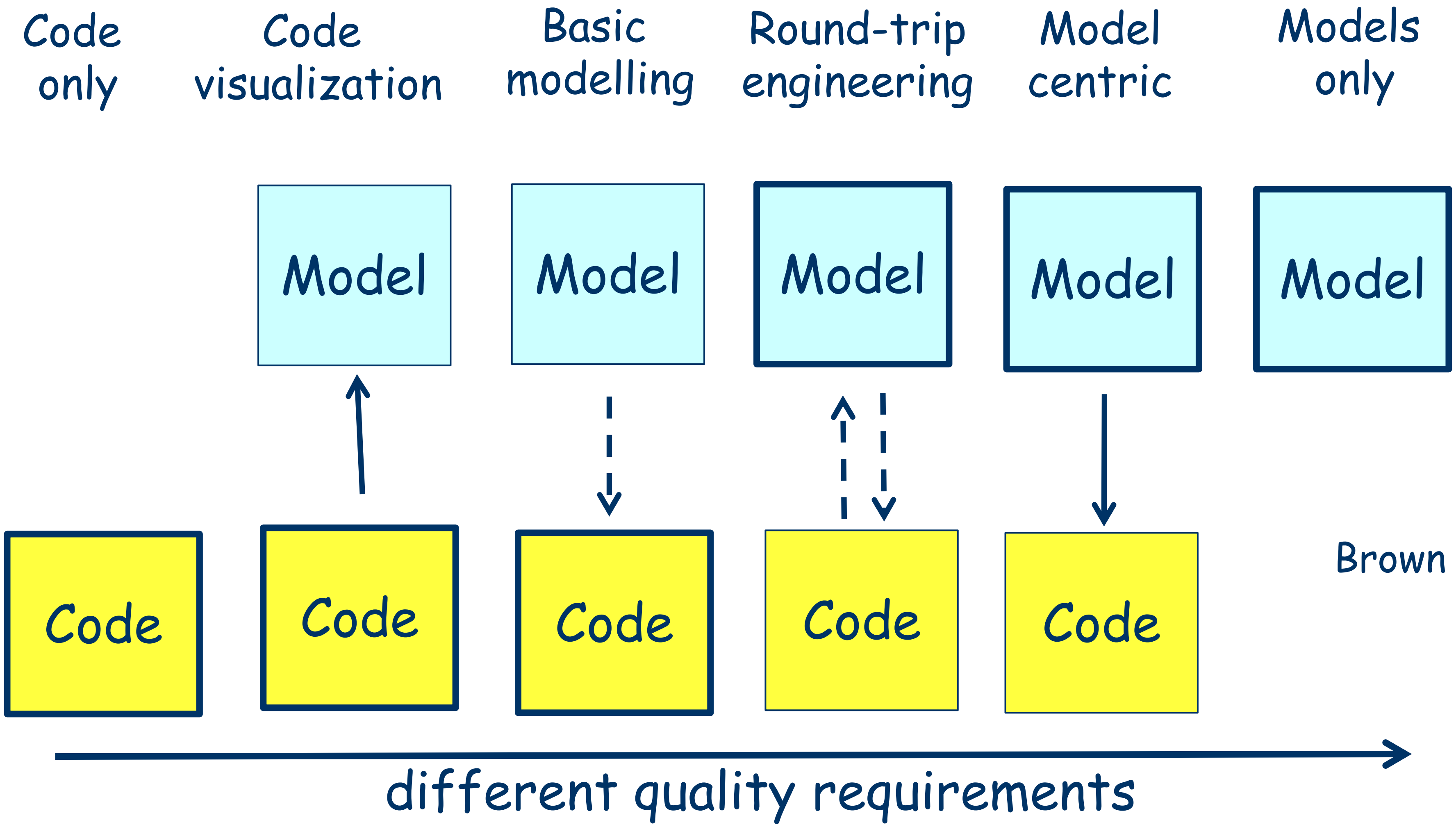
- Models as **communication means**
  - To communicate our understanding of a system to others
    - abstract, comprehensible
- Models as **blueprints**
  - To specify how a system should be implemented
    - correct, complete (??) and consistent with one another
- Models as **analysis and design tool**
  - To analyse a system and its environments
  - To predict some characteristics of the system
    - Predictive models, executable models
- Models as "**the system**"
  - To generate (most of the) source code from models
    - modifiable, manageable, cost-effective, compositionable, …

Model-driven engineering

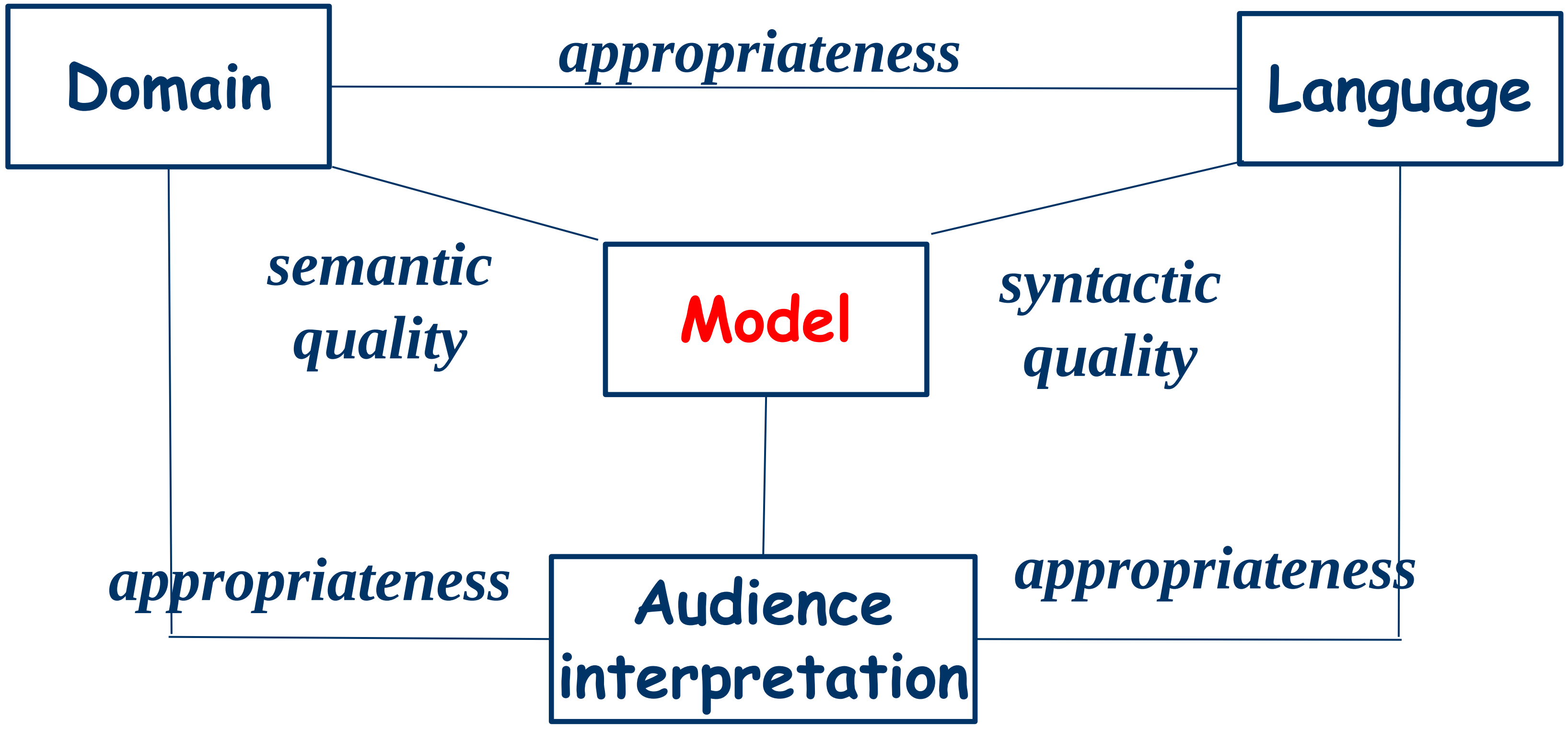
"Model" in this context

- A description or representation of a software system or its environment for a **certain purpose**, developed using a **modelling language** and thus conforming to a metamodel.
- A model may consist of several **diagrams** where each diagram type gives a different view on the described system.
  - For example UML 2.0 has 13 diagram types such as use case diagram, class diagram etc.
- It is common to model a software system at different abstraction levels.
- For each purpose of modelling, a suitable language is important.

The spectrum of modelling



Lindland et al.'s quality model for conceptual models (1994)



Syntactic quality -> syntactic correctness
Semantic quality -> validity and completeness
Pragmatic quality -> comprehension

How to define model quality?

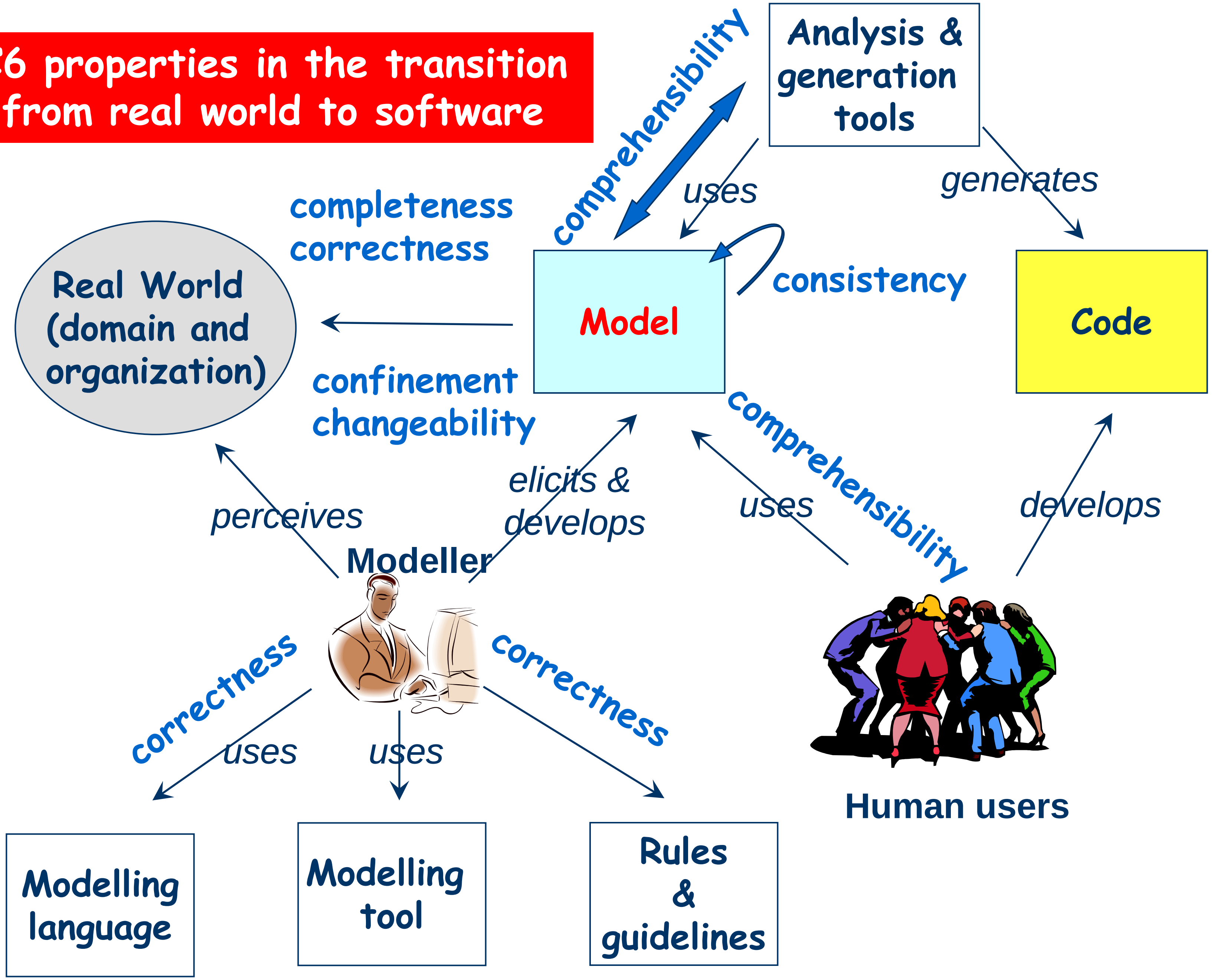
- Model quality has different aspects or may be defined by different characteristics;
- Definitions of such characteristics should be easy to grasp;
- It should be possible to evaluate a quality characteristics;
- We are interested in covering aspects important in model-centric or model-driven engineering.

- We performed a review of literature to extract what model quality means in practice!

The C6 properties

- **Correctness**
  - correct elements and correct relations between them
  - not violating rules and conventions
- **Completeness**
  - having all the necessary information and being detailed enough
- **Consistency**
  - no contradictions in the models
- **Comprehensibility**
  - being understandable by the intended users, either people or tools
- **Confinement**
  - being in agreement/restricted with the purpose of modeling and the type of system, right abstraction level
- **Changeability**
  - supporting changes or improvements with minimal necessary effort
  - Supporting modularity and composition

C6 properties in the transition from real world to software



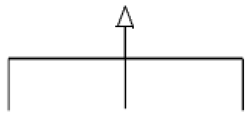
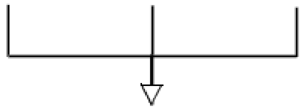
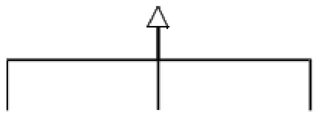
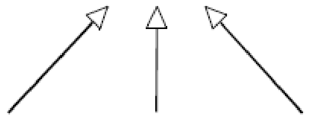
Improving the quality of models during development

- Error prevention or quality by construction
  - Modelling conventions; Do's and Don'ts
  - Iterative development; Agile modelling
  - Tools (by monitoring)
    - Constraints on model element
  - Using languages close to the domain
  - Generating models from other models
    - Quality-driven transformations
  - Formal models
    - Using formal languages
    - Combining UML with other languages

Modelling conventions

- "guidelines for creating effective (UML) diagrams; based on proven principles that will lead to diagrams that are easier to understand and work with". Ambler
  - a class with a high number of outgoing relations indicate that the class depends on too many other classes
  - Every actor in the model should communicate with use cases through interfaces
- Classification (Lange et al.):
  - Design conventions
  - Syntax conventions
  - Diagram conventions
  - Application domain-specific conventions

Example of diagram conventions

Notational Difference	Variation (a)	Variation (b)
Inheritance direction (N1)	 (Page-Jones 2000)	 (Purchase, Allder & Carrington. 2000)
Inheritance arcs (N2)	 (Page-Jones 2000)	 (Rumbaugh et al. 1999)

Quality assurance once the models are developed

- Error detection
  - automatically or manual
- Model reviews
  - Find defects, analyse fit for purpose, involve experts
- Tools (by analysis)
  - naming conflicts, missing elements, incorrectly defined interfaces, other rules
- Model checking for formal models
  - OCL evaluator, SPIN model checking
- Measuring models

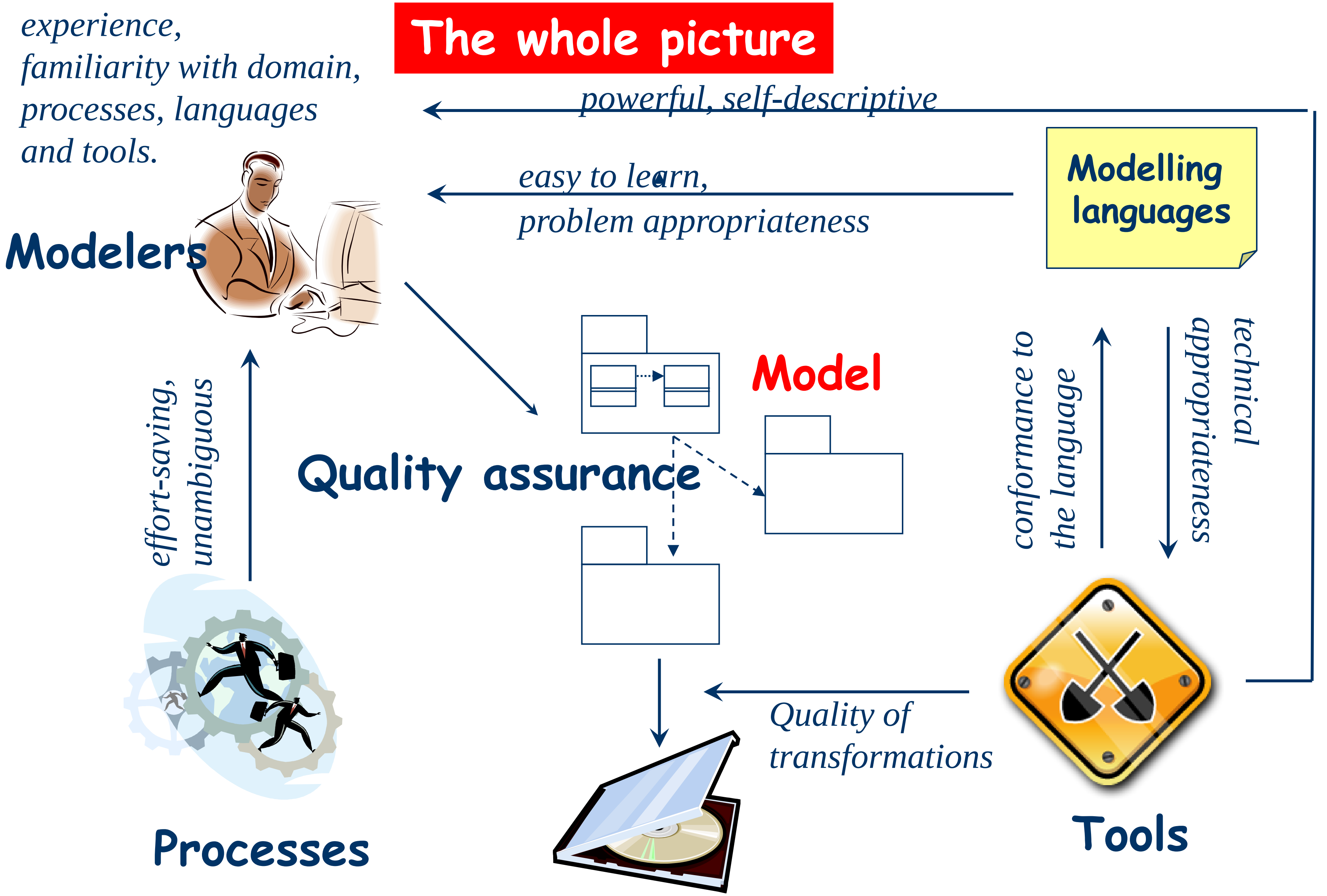
Measuring models

- Models for capturing and communicating a system's specification or main characteristics:
  - completeness of requirement models
  - correspondence between a model and the problem domain
  - Time it takes to understand a model or do some changes
- Models for design and implementation
  - Object-Oriented design metrics
  - Size metrics
    - counting the elements contained within a model; for example, the number of operations in a class, the number of classes in a package, the number of use cases

Advantages of metrics

- Early evaluation
  - Size of a system, its complexity
- Implementation language independence
  - Source code metrics are language dependent while model metrics are not
  - The possibility to evaluate some characteristics both before and after adding implementation details, such as dependencies between the elements of a model
- Prediction
  - Cost, development time
  - Monitor bottlenecks
    - Performance engineering models in MODELPLEX

The whole picture


experience, familiarity with domain, processes, languages and tools.
Modelers
powerful, self-descriptive
easy to learn, problem appropriateness
Modelling languages
Model
Quality assurance
effort-saving, unambiguous
conformance to the language
appropriateness
technical
Processes
Quality of transformations
Tools

Some challenges

- Assessing quality has two parts:
  - Measuring
    - Metrics has mostly been defined for the low-level structural design models or size metrics.
  - Judgement
    - what is good or bad?
    - What is the baseline data?

- Better modelling tools:
  - Tools can facilitate developing high-quality models regarding consistency, aesthetics, syntactical correctness
- Quality of modelling languages and modelling processes

Summary

- Model-based software development can improve the quality of software and mitigate important risks
  - Prediction, improving design, reduced dependency on underlying platforms, automated generation
  - Modelling also at business level is gaining popularity
- High-quality models can improve the quality of software, even for non full model-driven projects.
- Research is still needed on developing proper quality goals and evaluation methods related to model quality.

References

- S.W. Ambler, The Elements of UML 2.0 Style, Cambridge University Press, 2005.
- B. Berenbach, Metrics for model driven requirements development, Proc. 28th International Conference on Software engineering (ICSE'06), 2006, pp. 445-451.
- R. Conradi, P. Mohagheghi, T. Arif, L.C. Hedge, G.A. Bunde, A. Pedersen, Object-Oriented reading techniques for inspection of UML models – an industrial experiment, in: Proc. the European Conference on Object-Oriented Programming (ECOOP'03), LNCS volume 2749, 2003, pp. 483-501.
- C.F.J. Lange, B. DuBois, M.R.V. Chaudron, S. Demeyer, An experimental investigation of UML modeling conventions, in: Proc. MODELS'06, 2006, pp. 27-41.
- O.I. Lindland, G. Sindre, A. Sølvberg, Understanding quality in conceptual modelling, IEEE Software 11-2 (1994) 42-49.
- P. Mohagheghi, V. Dehlen, T. Neple, Definitions and approaches to model quality in model-based software development – a review of literature. Journal of Information and Software Technology - Quality of UML Models, Volume 51, Issue 12 (2009), pp. 1646-1669.
- P. Mohagheghi, V. Dehlen, Existing Model Metrics and Relations to Model Quality. ICSE Workshop on Software Quality 2009, WOSQ '09.
- H.C. Purchase, L. Colpoys, M. McGill, D. Carrington, C. Britton, UML class diagram syntax: an empirical study of comprehension, Proc. Australian Symposium on Information, 2001, pp. 113-120.

Thank you

Questions?

Comments?

More on our research in http://quality-mde.org/