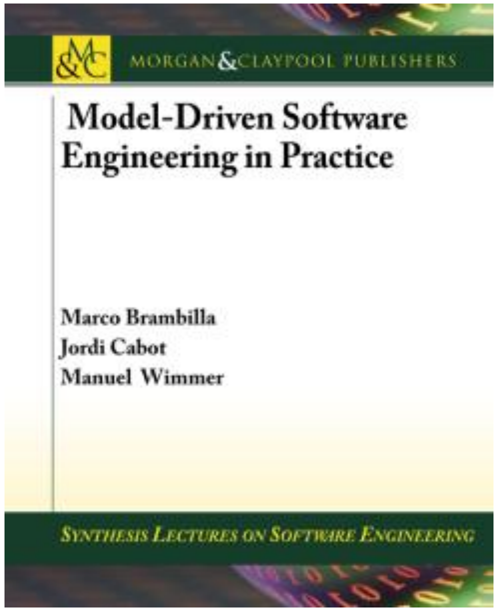
Chapter #5

INTEGRATION OF MDSE IN YOUR DEV. PROCESS

Teaching material for the book
Model-Driven Software Engineering in Practice
by Marco Brambilla, Jordi Cabot, Manuel Wimmer.
Morgan & Claypool, USA, 2012.


MORGAN & CLAYPOOL PUBLISHERS
Model-Driven Software Engineering in Practice
Marco Brambilla
Jordi Cabot
Manuel Wimmer
SYNTHESIS LECTURES ON SOFTWARE ENGINEERING

Content

- General advice
- MDSE in a *traditional* development process
- Agile MDSE
- Domain-driven design and MDSE
- Test-driven development and MDSE

MDSE IN YOUR DEV. PROCESS: THINGS TO CONSIDER FIRST

Morgan & Claypool Publishers
Model-Driven Software Engineering in Practice
Marco Brambilla
Jordi Cabot
Manuel Wimmer
Synthesis Lectures on Software Engineering

Technological adoption

Why Model Engineering?

- In any change of technology, organizational, managerial and social aspects are the main reasons of failure
- Introducing MDSE without considering these aspects is a sure path to failure
- Some common-sense advice:
  - First MDSE project should not be a critical one
  - Make sure management is committed
  - Get somebody with experience on board
  - Start small, with a pilot project and grow from there

Socio-technical aspects

Why Model Engineering?

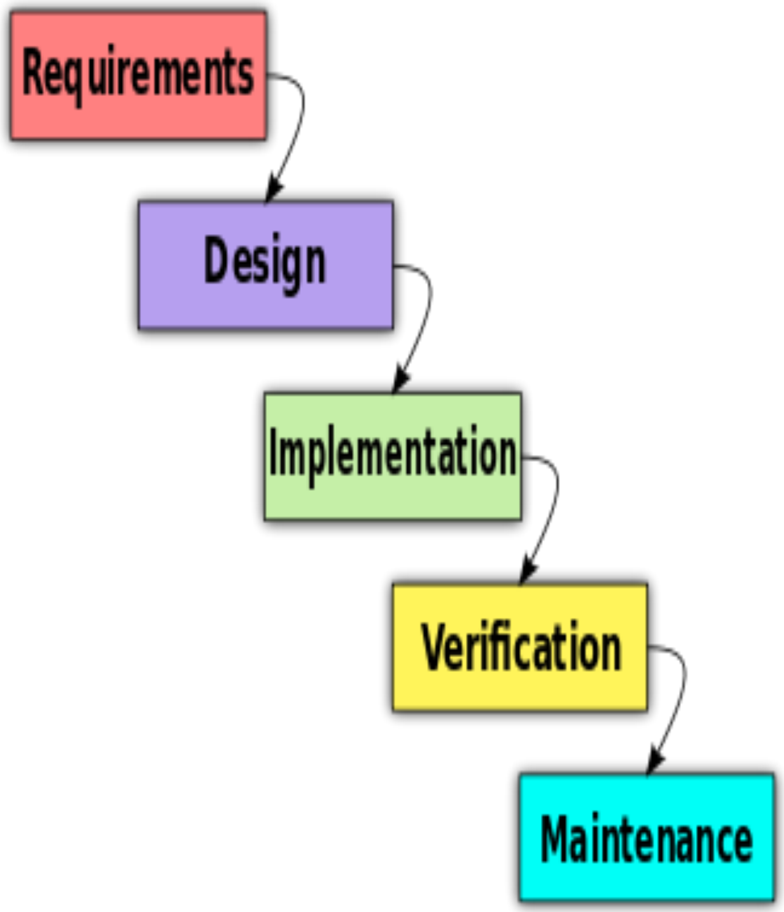
- **Pains and gains of software modeling**
  - Modeling introduces new tasks and roles in the dev. Process
  - Some of them are a pain (i.e. now there is more work to be done)
  - Some others get the gain (i.e. maintenance is easier with models)
  - If people in the pain and the gain sides are not the same be careful with motivation and perception problems on the use of modeling. Recognize the *pain* work

- **Socio-technical congruence**:
  - MDSE requires new skills, roles and dependencies in the dev. team
  - Your organization must be able to match those requirements (e.g. if nobody enjoys / is good at modeling, who will take in charge the modeling tasks in the process?).

MDSE IN A TRADITIONAL DEVELOMENT PROCESS

Classical development processes

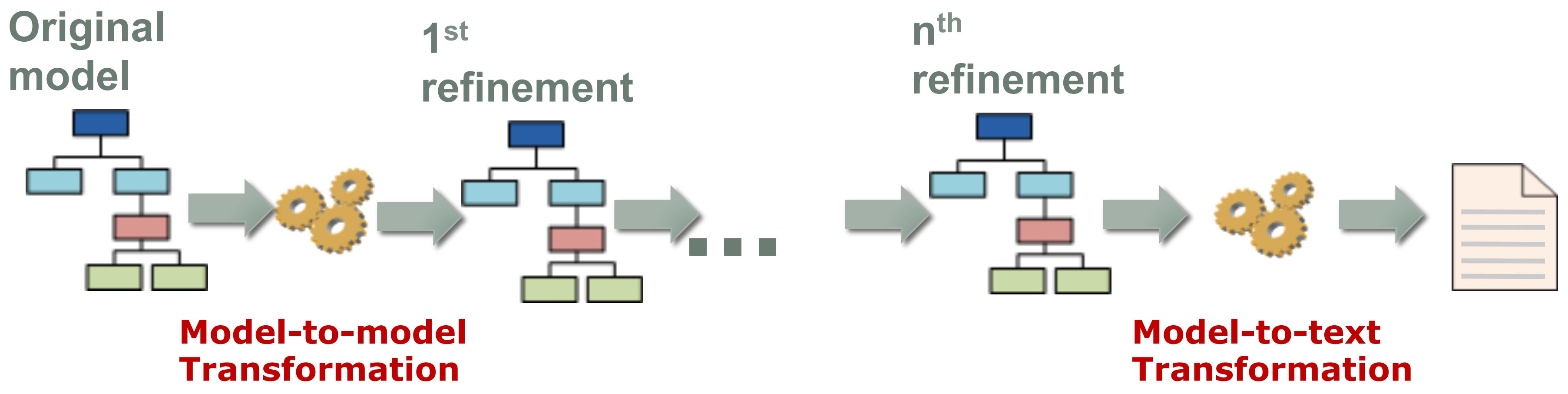
Waterfall, spiral, iterative, incremental, UP….



- Already model-based.
- Models are typically employed in each phase of the process
  - Requirement models
  - Analysis models
  - Design models
  - Deployment models…
- How MDSE contributes?

MDSE in Classical dev. processes

- Key contribution: Going from model-based to model-driven
  - Opportunity to (semi)automate the transitions between the different phases of the process



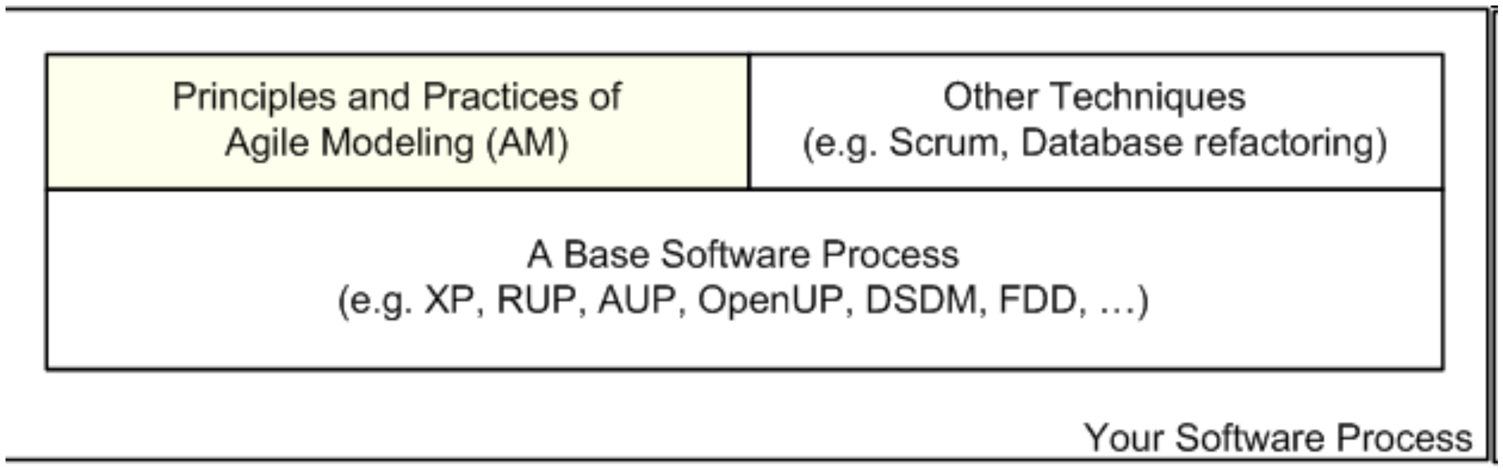
HAS MDSE A PLACE IN AN AGILE WORLD?

Agile development process

- Agile Manifesto proposes to center development around:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
- Has MDSE a place in this manifesto? Common criticism:
  - Models are not working software
  - Can’t be tested
  - Are just documentation
  - Extra work to adapt to changes
- But we know better (e.g. models are executable) and others agree…

Agile Modeling

- Collection of modeling principles and practices suited for lightweight development processes. Lead by Scott W. Ambler
- Goal: avoid modeling for the sake of modeling


Principles and Practices of Agile Modeling (AM)
Other Techniques (e.g. Scrum, Database refactoring)
A Base Software Process (e.g. XP, RUP, AUP, OpenUP, DSDM, FDD, …)
Your Software Process

Copyright 2001-2006 Scott W. Ambler

Agile Modeling

Principles (I)

- **Model With A Purpose**. identify a valid purpose for creating a model and the audience for that model, then develop it to the point where it is both sufficiently accurate and sufficiently detailed.
- **Travel Light**. Every artifact that you create, and then decide to keep, will need to be maintained over time. Trade-off agility for convenience of having that information available to your team in an abstract manner.
- **Multiple Models**. You need to use multiple models to develop software because each model describes a single aspect/view of your software.
- **Rapid Feedback**. By working with other people on a model you are obtaining near-instant feedback on your ideas.
- **Assume Simplicity**. Keep your models as simple as possible. Don't depict additional features that you don't need today. You can always refactor in the future (yes, there are model refactoring techniques)

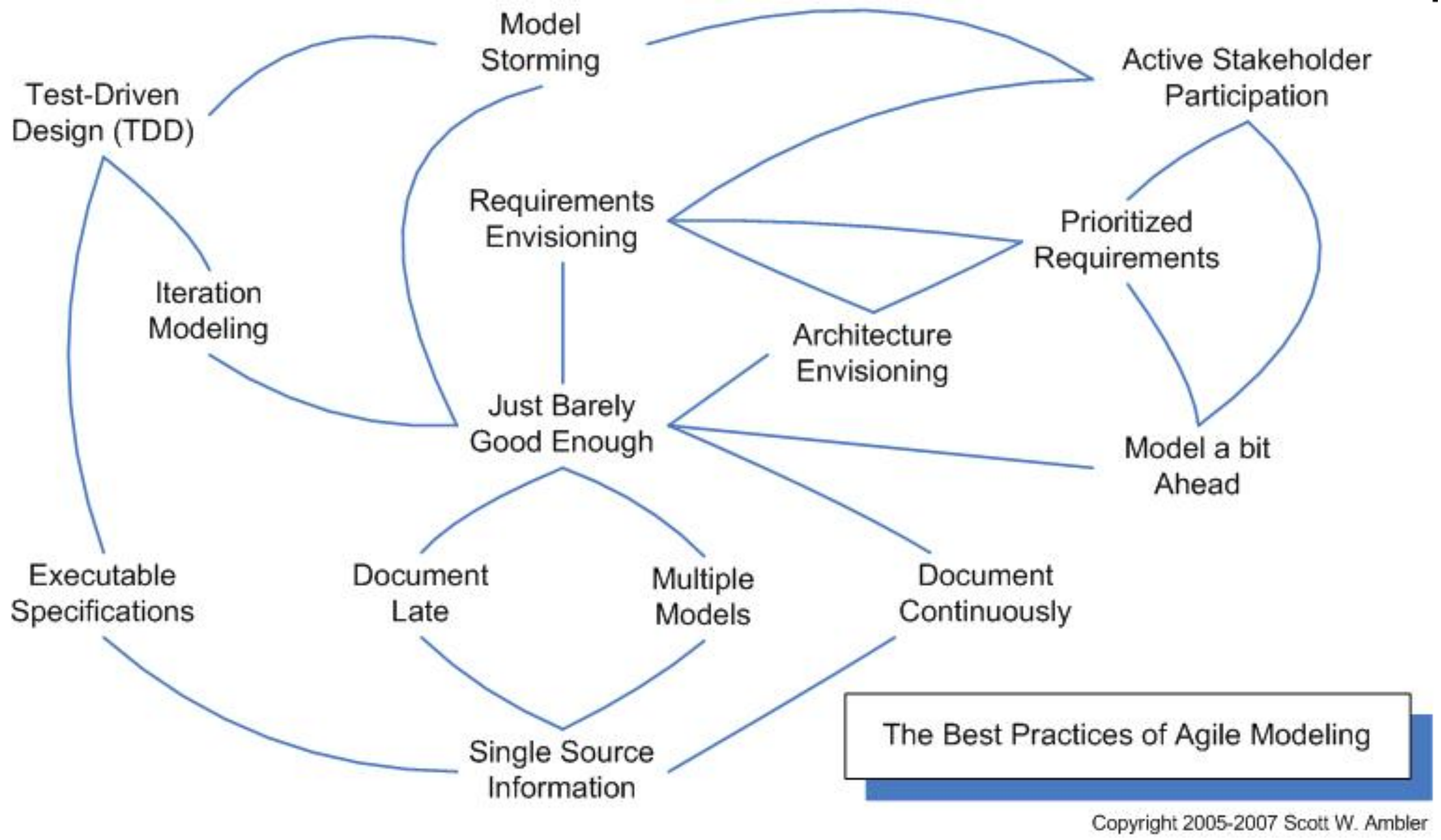
Agile Modeling

Principles (II)

- **Embrace Change**. Requirements evolve over time and so your models
- **Incremental Change**. Develop good enough models. Evolve models over time (or simply discard it when you no longer need it) in an incremental manner.
- **Working Software Is Your Primary Goal**. The primary goal is not to produce extraneous documentation, extraneous management artifacts, or even models. Any (modeling) activity that does not directly contribute to this goal should be questioned
- **Enabling The Next Effort Is Your Secondary Goal**. To enable it you will not only want to develop quality software but also create just enough documentation and supporting materials so that the people playing the next game can be effective.

Agile Modeling

Practices



Agile MDSE

- Agile Modeling + executable models
- Effective modeling of executable models to go from models to working software automatically in the most agile possible way.

MDSE VS DOMAIN-DRIVEN DESIGN

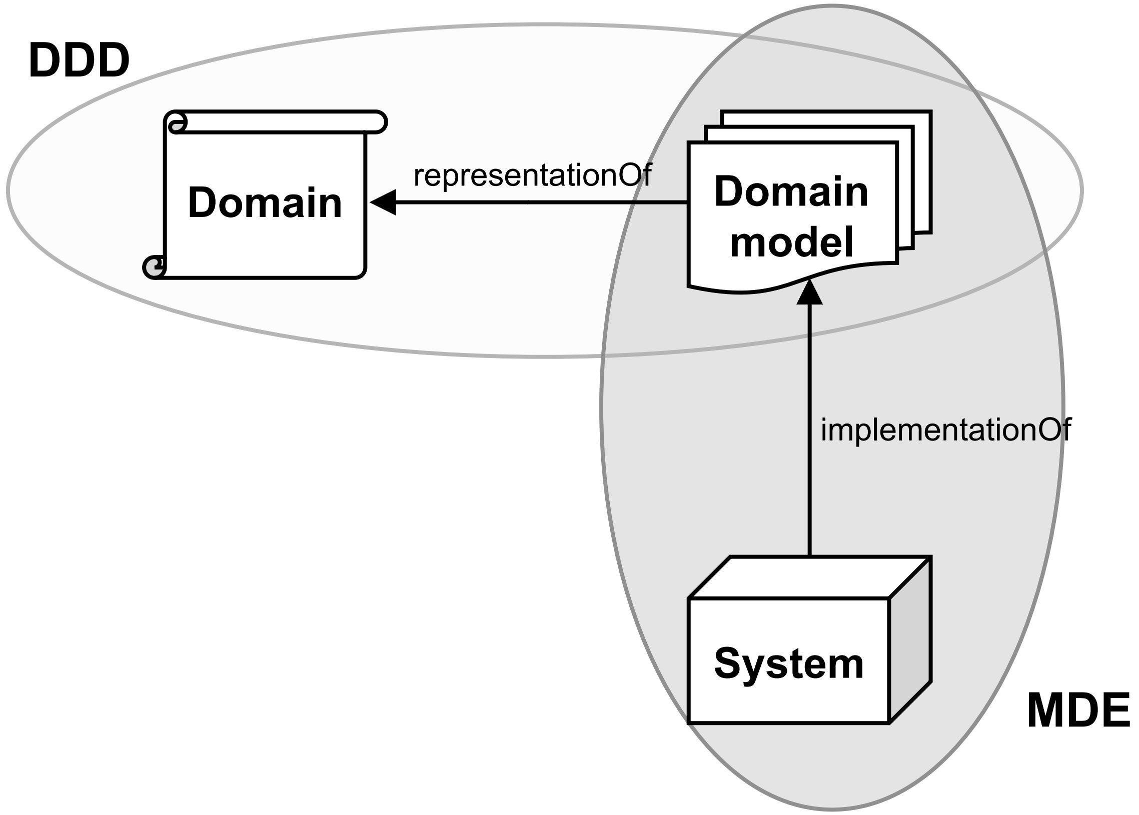
Domain-driven design

- Domain-driven design (DDD) is based on two main ideas:
  - The primary focus of a SW project should be the domain itself and not the technical details
  - Complex domains must be modeled first. A set of design practices is provided to create these models.
- Thus, DDD emphasizes the importance of domain models.

- DDD and MDSE have commonalities:
  - Need of using models to represent the system domain
  - Focus on platform-independent aspects (using DMA terminology)

Domain-driven design

- MDSE in DDD:
  - Provides a framework to put DDD in practice (e.g. by providing modeling languages that can be used in DDD)
  - Maximizes the benefit you can get out of the domain models (e.g. by transforming them into running code)



MDSE AND TEST-DRIVEN DEVELOPMENT

Model-Driven Software Engineering in Practice

Marco Brambilla
Jordi Cabot
Manuel Wimmer

Test-driven development (TDD)

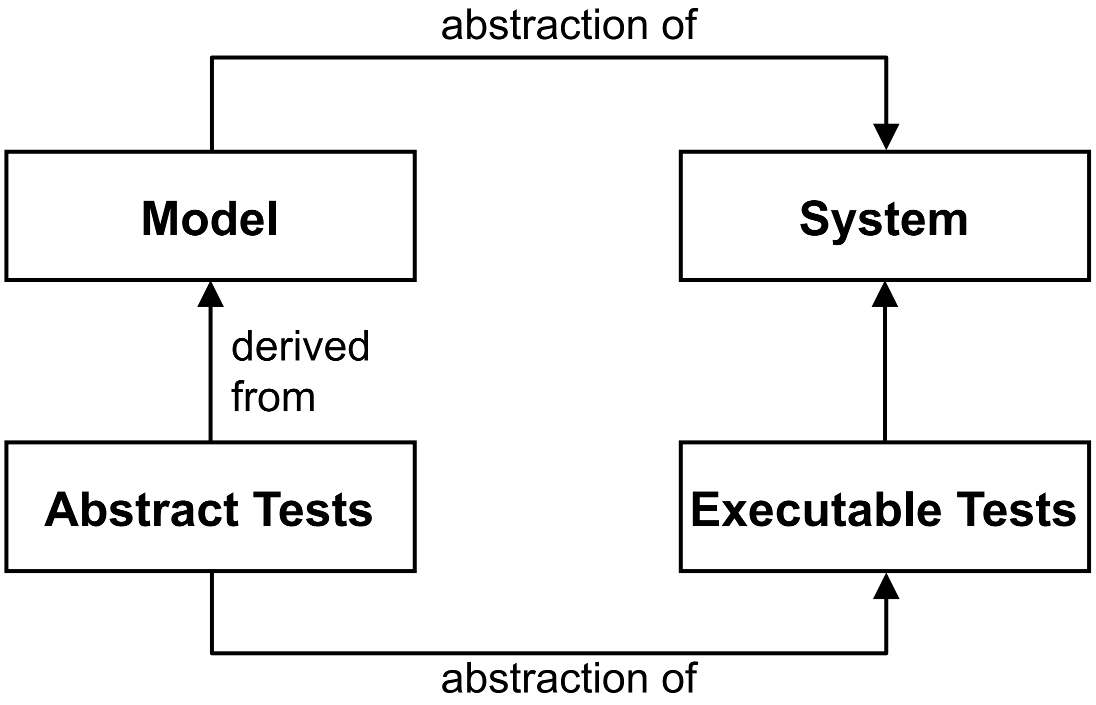
- Test-first philosophy:
  - Create an executable test to check the correctness of the new functionality-to-be
  - Develop the code to pass the test
  - Refactor the code and repeat
- Integration of MDSE in TDD can happen at two different levels, depending on the kind of MDSE process we follow
  - Model-driven testing
  - Test-driven modeling

Model-driven testing

Derive tests from your models

- If the system is NOT automatically generated from the models, we need to check the implementation behaves as expected (i.e. as defined in the models).

- Models can be used to generate the tests that the implementation will need to pass.



Test-driven modeling

Test-first your models

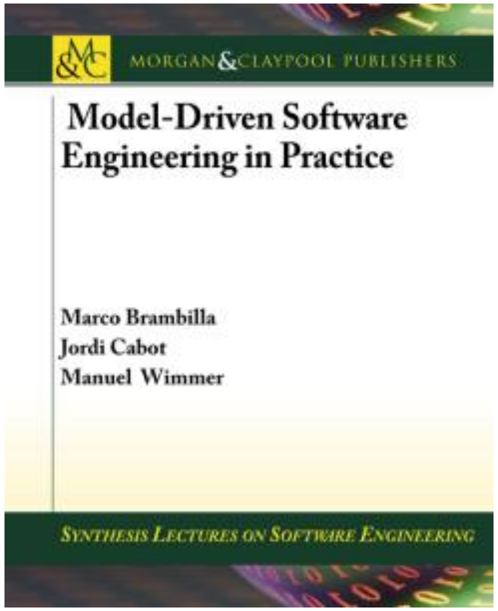
- If the system is automatically generated from the models then there is no need to test the system.
- Models should be then the focus of your testing strategy.
- For each new model excerpt, write first the model test, then write the model and check the model passes the test
  - In the *Model Management* chapter you'll see some model validation and verification tools that can be used for this purpose

MORGAN & CLAYPOOL PUBLISHERS

MODEL-DRIVEN SOFTWARE ENGINEERING IN PRACTICE

Marco Brambilla,
Jordi Cabot,
Manuel Wimmer.
Morgan & Claypool, USA, 2012.

www.mdse-book.com
www.morganclaypool.com
or buy it at: www.amazon.com


MORGAN & CLAYPOOL PUBLISHERS
Model-Driven Software Engineering in Practice
Marco Brambilla
Jordi Cabot
Manuel Wimmer
SYNTHESIS LECTURES ON SOFTWARE ENGINEERING