Model-Driven Development

01. Course Introduction

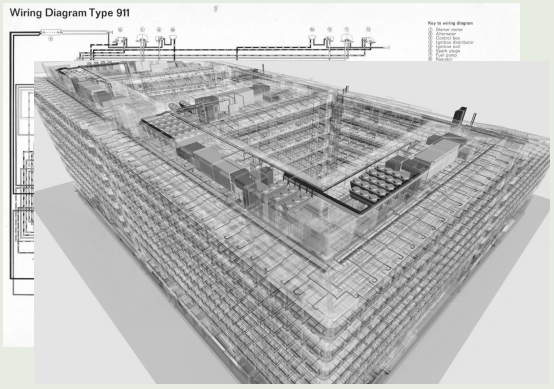
Andrzej Wąsowski

IT UNIVERSITY OF COPENHAGEN

PROCESS AND SYSTEM MODELS GROUP

Modeling in Engineering

Models are Abstractions

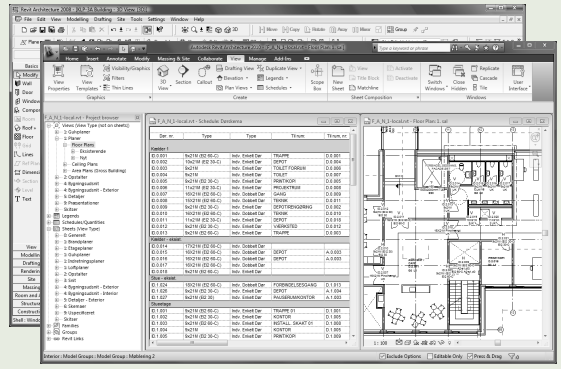


*Models help us **understand** (...) potential solutions through abstraction. (...) Software systems, which are often among the most complex engineering systems, can benefit greatly from using models*[†].

Software Systems

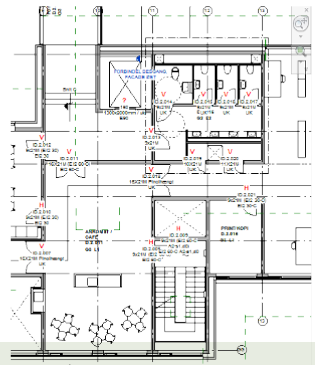
- more complex than other systems
- Boeing-747: **6** milion parts, incl. **3** mio. fasteners
- Linux: **15** MLOC, OpenOffice **9** MLOC, Windows **50** MLOC (already in 2003!)

Digital Models



Avionics
Automotive

Construction Eng.
Environmental Eng.
Mechatronics
Control



abstraction combats complexity

computers enable automation

[†]Bran Selic. *The pragmatics of model-driven development*. IEEE Software, 20(5):19–25, 2003

Example 1: Yahoo Pipes

a webapp for building data mash-ups that aggregate web feeds, web pages, and other services.

a domain specific language for end users

a visual language

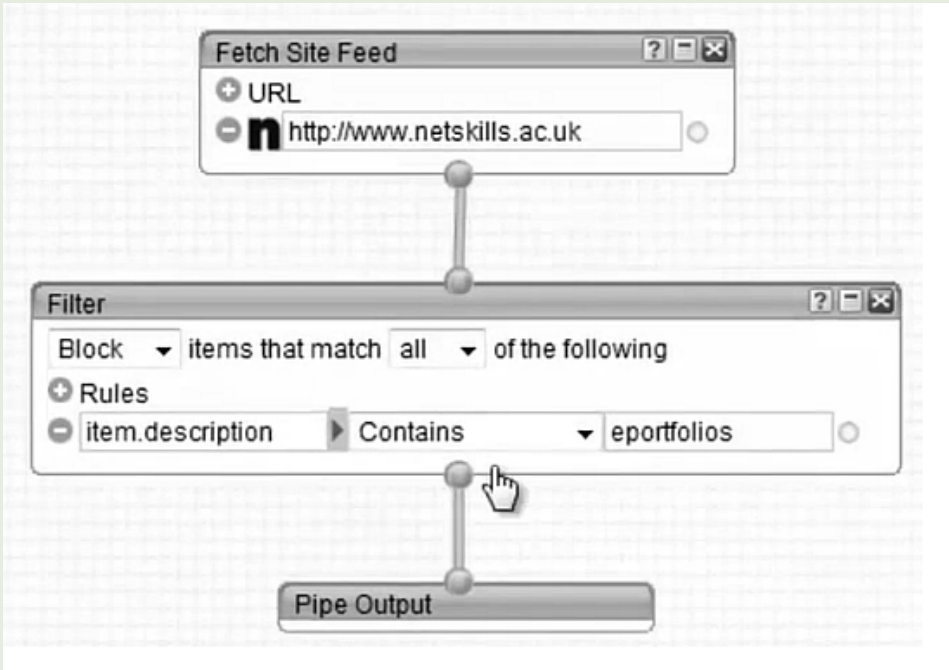
online DSL editor

likely interpreted,
with independent offline compilers

familiar abstractions:
loops, conditions, events ...

a control flow language,
but still very much unlike Java

Yahoo Pipes†



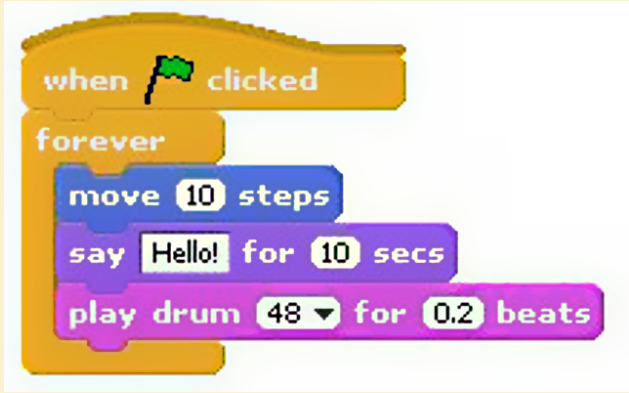
language at very high level of abstraction

no protocols, algorithms, structures, parsing...

a data-flow language, very different from, say, Java ...

MIT Scratch*

A language for teaching programming to children through play. Allows creating animations and games.



CC-BY-SA andresmh

tutorial video:http://www.youtube.com/watch?v=Uu-D1DCXaQw

†http://pipes.yahoo.com (since 2007)

*http://scratch.mit.itu/ (since 2003, milion+ users)

Google Protocol Buffers

a flexible, efficient, automated mechanism for serializing and persisting structured data;
initially developed for index server request/response protocol†

Example 2: a textual DSL for developers

person.proto

```
1  message Person {
2    enum PhType { MOBILE = 0; WORK = 1; }
3    message PhoneNo {
4      required string no = 1;
5      optional PhType type = 2 [default=HOME];
6    }
7
8    required string name = 1;
9    repeated PhoneNo phone = 2;
10 }
```

interface description lang.

structural modeling lang.

smaller and faster than XML

unreadable, use txt to debug

protocol buffers support

messages, attributes,
enumerations,
cardinalites,
field numbering,
defaults values,
simple types,
hierarchical nesting,
but **no** references.

using generated code in Java

read person in python or C++!

```
11 Person.Builder person = Person.newBuilder();
12 person.setName("Alice");
13 Person.PhoneNo.Builder phoneNo =
14    Person.PhoneNo.newBuilder().setNo("7258-00");
15 person.addPhone(phoneNo);
16 person.build().writeTo(/*output-stream-here*/);
```

python™

Java

C++

TXT

.proto is compiled

data access,
serialization,
deserialization
automatically
generated

†https://developers.google.com/protocol-buffers/
48,162 message types in Google code tree in 12,183 .proto files [developers.google.com, Feb'14]

Example 3: Ruby on Rails

a case for automation and code generation

- **Ruby** is a dynamically typed, interpreted, object-oriented language
- **Ruby on Rails** is a web application framework implemented in Ruby
  - gathers information from the web server and database
  - renders web pages
- Follows established conventions facilitating rapid development:
  - active record pattern
  - convention over configuration (CoC)
  - don't repeat yourself (DRY)
  - model–view–controller (MVC) pattern

- Rails uses **code generation**
- Let's peek [until 3:50] http://www.youtube.com/watch?v=Gzj723LkRJY

Rails use an internal DSL

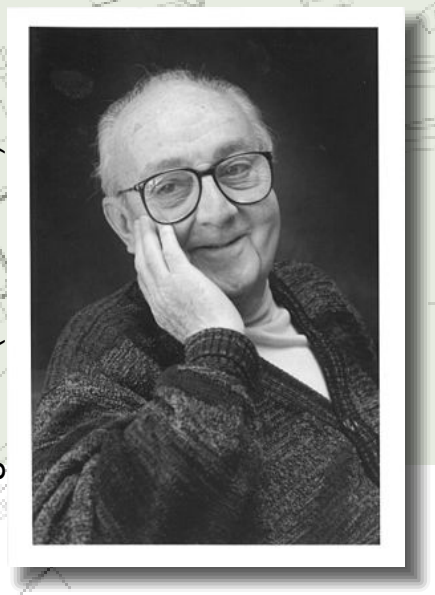
```
class Client < ActiveRecord::Base
  has_one  :address
  has_many :orders
  has_and_belongs_to_many :roles
end
```

What is a Model?

Definition. A model is an **abstraction** of reality, of a system, made with a specific **purpose** in mind.

All models are wrong, but some are useful[†]

Incidentally statistics is another discipline that works with models.

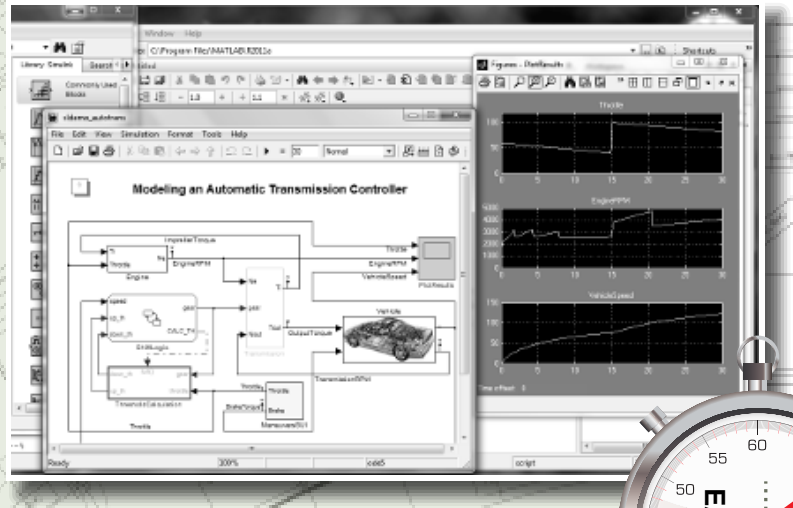


George Box (1919-2013)

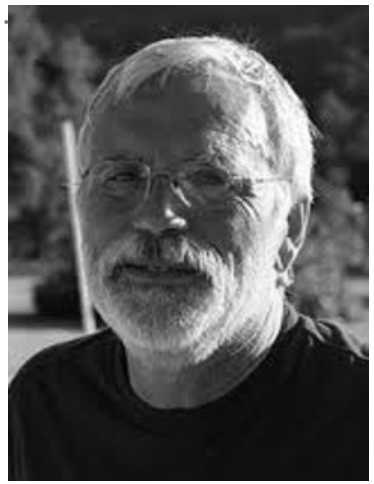
[†]George Box. Norman Draper. *Empirical Model-Building and Response Surfaces*. p. 424, Wiley, 1987

What is Model-Driven Development? Why?

Model-driven software development is a software development methodology which focuses on creating and exploiting domain models (that is, abstract representations of the knowledge and activities that govern a particular application domain)†.



MDD's defining characteristics is that software developments primary focus and products are models rather than computer programs. [...] A key characteristic of these methods is their fundamental reliance on automation and the benefits that it brings*.



Bran Selic

What for?

- **design** SMDP
- **code generation** SMDP
- simulation
- testing, verification
- runtime monitoring
- doc. generation
- performance analysis

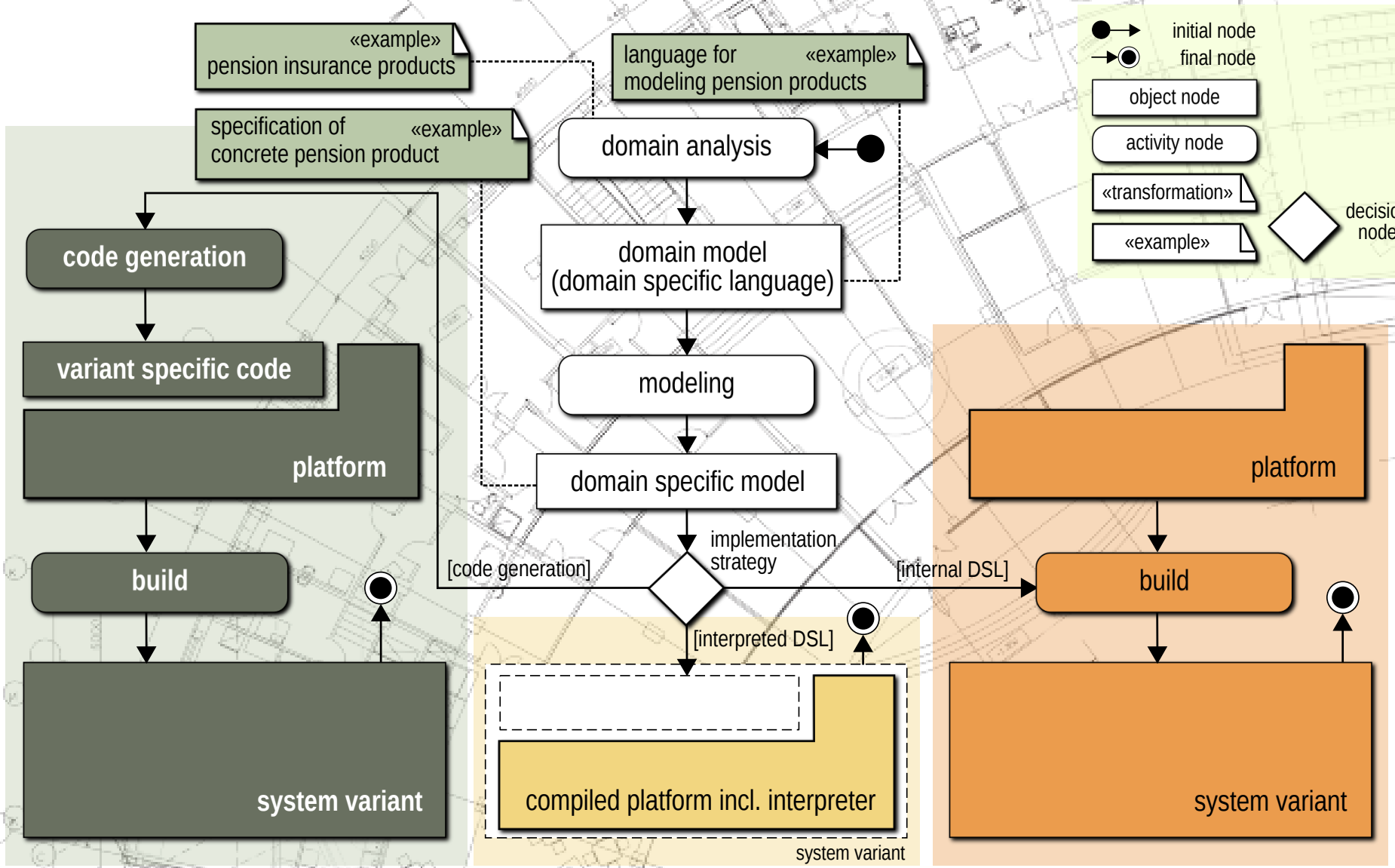
†Wikipedia, MDE, 2011/08/27

*Bran Selic. *The pragmatics of model-driven development*. IEEE Software, 20(5):19–25, 2003

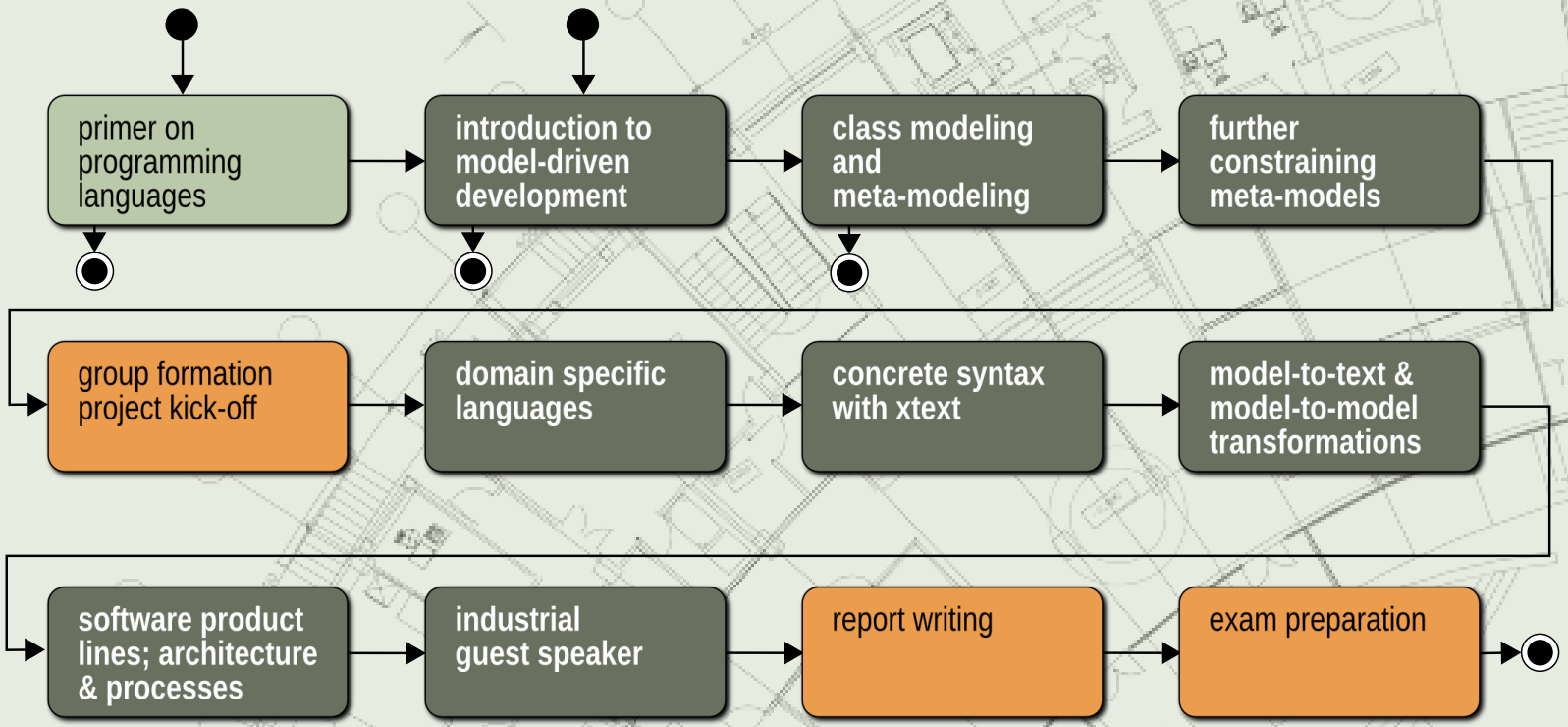
MATLAB SIMULINK

Typical DSL Implementation Architectures

Three scenarios



Course Overview



- How to model and why?
- Use domain modeling for understanding complex projects
- Process models automatically to transform or analyze them
- Generate code?
- Lots of hands-ons experience; legacy code and re-engineering
- 50% effort goes into project

A Few Words about Exam

- Format: oral (individual) + project report (group)
- You present the project briefly [2-3 minutes]
- We ask questions about it
- Conversation converges on some lecture topics [15 minutes]
- We can ask about any course material
- Juicy places marked with ⚠
  in the lecture notes
- Exam preparation class