



Domain-Specific Languages

Introduction

Andrzej Wąsowski

- 
- ▶ Modeling in (software) engineering
 - ▶ Model-driven software development
 - ▶ Implementing **domain specific languages**
 - ▶ Course overview
 - ▶ A few words about the exam

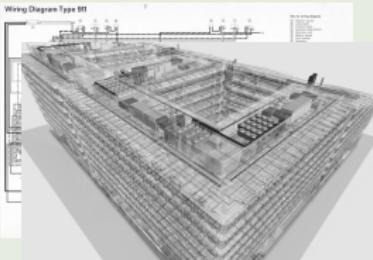


AGENDA

Modeling in Engineering

Digital Models

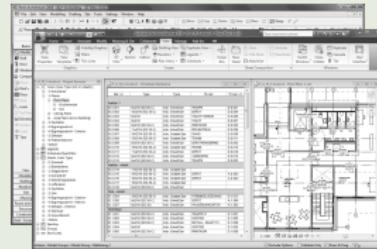
Models are Abstractions



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!)



Construction Eng.
Environmental Eng.
Mechatronics
Control

Avionics
Automotive



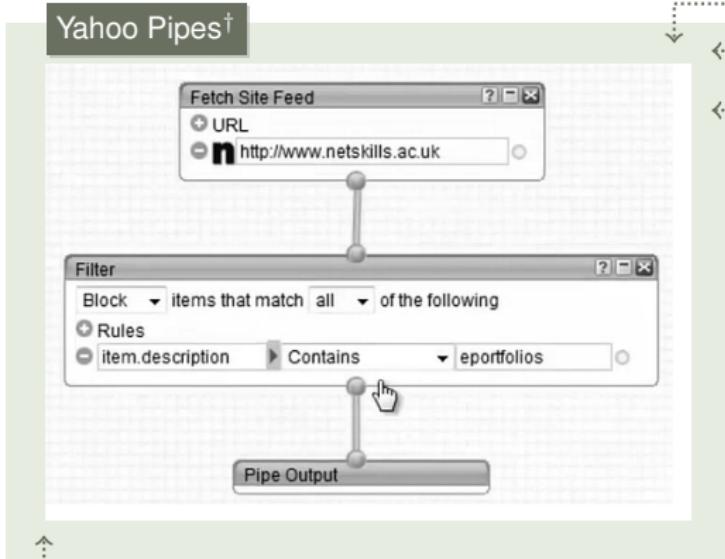
abstraction combats complexity

computers enable automation

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

Example 1: Yahoo Pipes

A domain specific language for end users



language at very high level of abstraction

no protocols, algorithms, structures, parsing...

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

A webapp for building data mashups that aggregate web feeds, web pages, and other services.

a visual language

online DSL editor

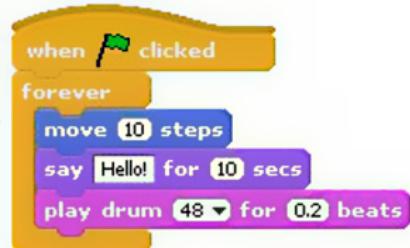
likely interpreted,
with independent offline compilers

familiar abstractions:
loops, conditions, events ...

a control flow language,
but still very much unlike Java

MIT Scratch*

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



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

*<http://pipes.yahoo.com> (since 2007)

<http://scratch.mit.edu> (since 2003, milion+ users)

CC-BY-SA andresmh

Google Protocol Buffers

Example 2: a textual DSL for developers

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

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=MOBILE];  
6   }  
7  
8   required string name = 1;  
9   repeated PhoneNo phone = 2;  
10 }
```

interface description lang.

structural modeling lang.

smaller and faster than XML

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*/);
```



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 [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 use an internal DSL

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

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

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.

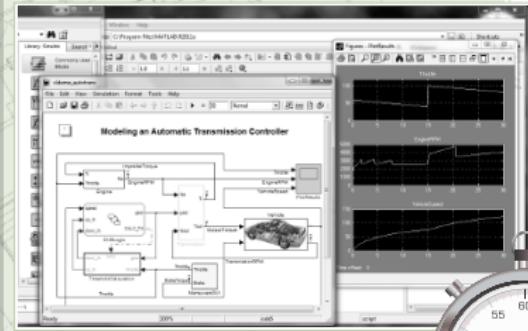


George Box (1919-2013)

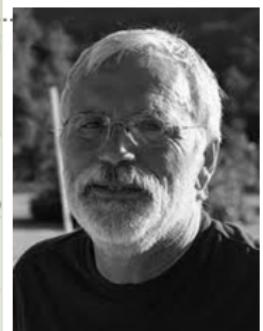
[†]George Box. Norman Draper. *Empirical Model-Building and Response Surfaces*. p. 424, Wiley, 1987

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

Assessment / Exam

- ▶ No mandatory homeworks
- ▶ Part 1: group project (semester hand-in)
- ▶ Part 2: oral with preparation time (individual)
- ▶ We deepen the answers in discussion
- ▶ We can ask about any course material
- ▶ Exam preparation class

