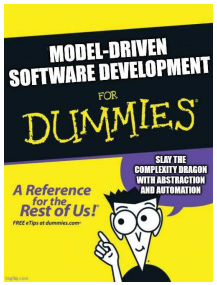
Beyond Code: An Introduction to Model-Driven Software Development (CISC 844)



Topic 0: Intro & Motivation, Overview, Admin

Juergen Dingel
Jan 2025

About Me

Small town Germany:
Born, raised, etc

Berlin: UG

Pittsburgh: PhD

Kingston: since 2000,
Formal methods,
Model-Driven Engineering,
SW Eng


A


B


C


D

You?

Modeling

- Models as purposeful abstractions
- Not a new idea…

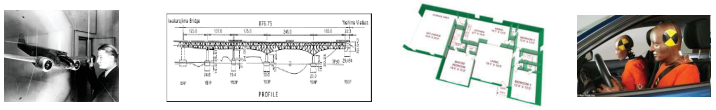
Modeling in other Disciplines

Natural sciences

Understanding, predicting existing phenomena ("Backwards Engineering")

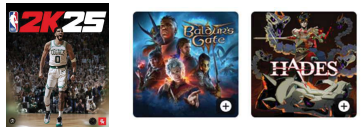
Engineering

Building artifacts with certain properties ("Forwards Engineering")



Entertainment

Doing what normally would be impossible

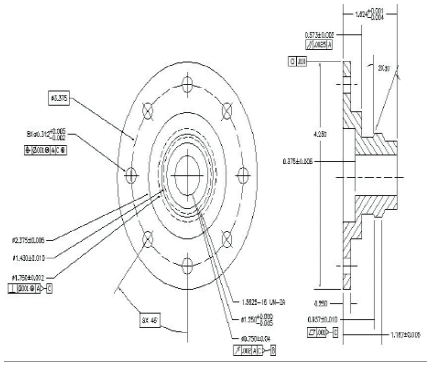


Modeling in Manufacturing

Mechanical design from 1800 to about 1980:

1. Draftsmen create 3-view drawings
2. Machinists create parts from drawings

⇒ laborious, error-prone, inefficient



Modeling in Manufacturing (Cont'd)

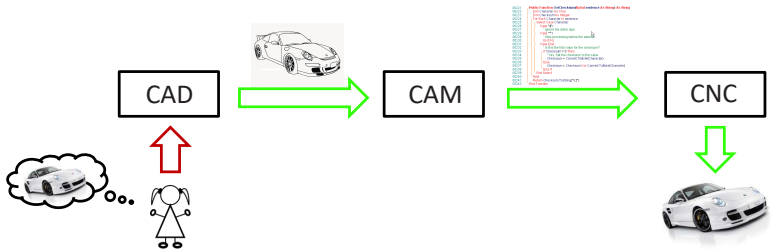
- **Example: Concorde (1976 – 2003)**
  - > 100,000 drawings
  - in 2 languages, using both metric and imperial systems

  ⇒ worked, but 7x over budget

Modeling in Manufacturing (Cont'd)

Mechanical design from about 1972: CAD/CAM

1. Create drawings w/ computer (CAD)
2. From drawing, computer automatically generates program to drive milling and CNC machines (CAM)



⇒ much better analysis capabilities and productivity

⇒ **CAD/CAM has revolutionized manufacturing**

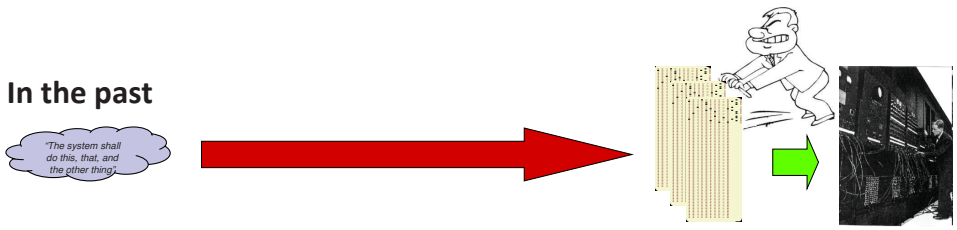
While Abstraction is Central to Computing…

- Stored-program concept
  Turing, Zuse, von Neumann ~ 1940
- Compilers and high-level languages
  Hopper, Backus ~ 1950
- Formal languages and automata
  Frege, Chomsky ~ 1956
- Time sharing
  Berner, McCarthy 1957
- Virtual Memory
  Fortheringham 1961, Kilburn et al 1962, Denning 1970
- Information hiding via modularization, encapsulation and interfaces
  Parnas, Hoare, Dahl ~ 1970

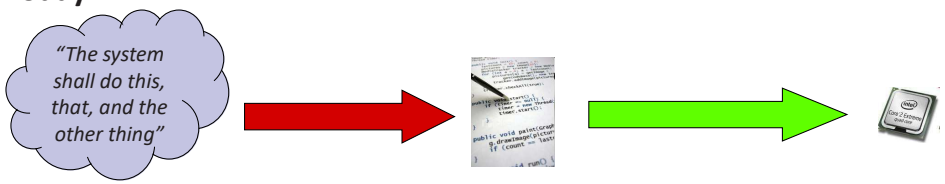
Search for "Influential Ideas in Computer Science"

While Abstraction is Central to Computing… (Cont'd)

In the past



Today



While Abstraction is Central to Computing… (Cont'd)

- **Abstraction**
  - Put more and more higher-level concepts into programming languages
  - Examples:
    - variables, basic data types
    - abstract data types (data abstraction)
    - functions and procedures (procedural abstraction)
    - objects
    - semaphores, locks

  but what makes this work in practice is

- **Automation**
  - automatically compile high-level concepts into executable code

… the Use of Models in an Engineering-like Way is Not!

Engineering

1. build (mathematical) models
2. analyze models rigorously
3. refine models
4. build artifact
5. little testing

Characteristics

- Very rigorous
- "front-loaded"
- **Main QA technique:** Modeling & analysis

Software Engineering

1. some (informal) modeling
2. build artifact
3. some (informal) reuse
4. lots of testing

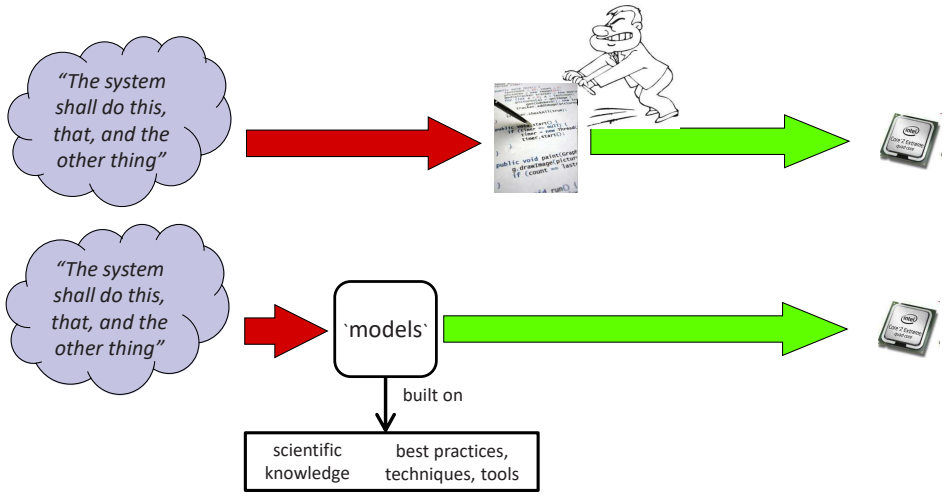
Characteristics

- Mostly informal
- "back-loaded"
- **Main QA technique:** Testing (often >50% of total development effort)

Making Software Engineering is considerably more `ad-hoc`!

What if… ?

… we made models the primary artifact in software engineering?



What if…?

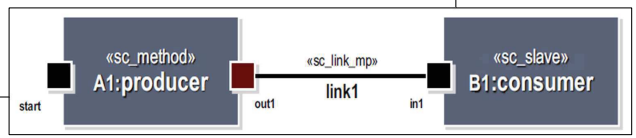
… the architecture of the system was not hidden in the code?

… design models were part of development?

[example courtesy Bran Selic]

```
SC_MODULE(producer)
{
sc_outmaster<int> out1;
sc_in<bool> start; // kick-start
void generate_data ()
{
for(int i =0; i <10; i++) {
out1 =i ; //to invoke slave;}
}
SC_CTOR(producer)
{
SC_METHOD(generate_data);
sensitive << start;}};
SC_MODULE(consumer)
{
sc_inslave<int> in1;
int sum; // state variable
void accumulate (){
sum += in1;
}
```

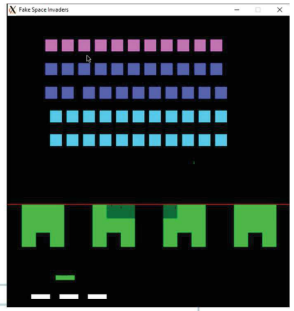
```
SC_CTOR(consumer)
{
SC_SLAVE(accumulate, in1);
sum = 0; // initialize
};
SC_MODULE(top) // container
{
producer *A1;
consumer *B1;
sc_link_mp<int> link1;
SC_CTOR(top)
{
A1 = new producer("A1");
A1.out1(link1);
B1 = new consumer("B1");
B1.in1(link1);}};
```

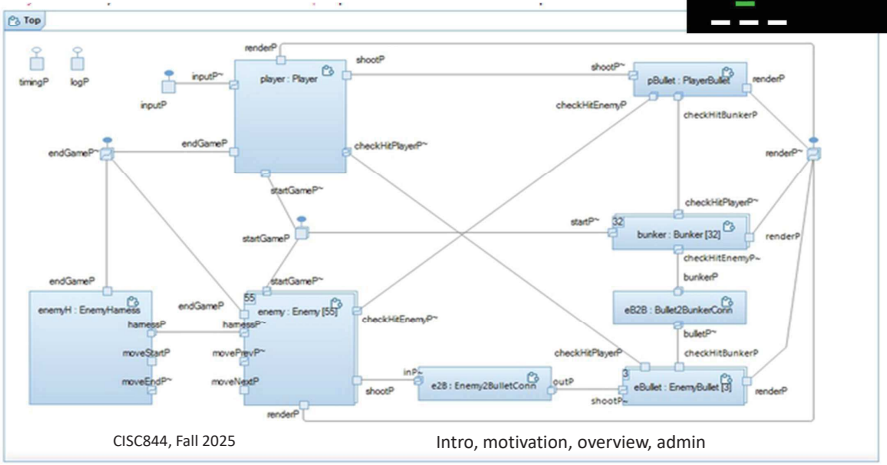


What if…?

… the architecture of the system was not hidden in code?

… design models were part of development?



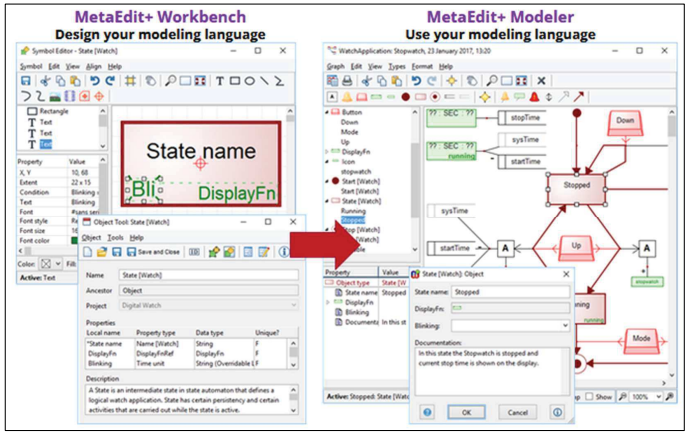


What if…?

… we could easily create languages & tooling specifically for a domain?

… rather than having to `encode` domain concepts in existing (textual) general purpose languages?

[www.metacase.com]

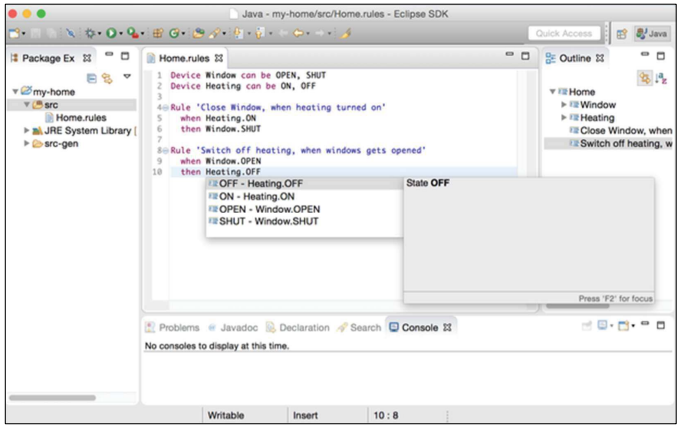


What if…?

… we could easily create languages & tooling specifically for a domain?

… rather than having to `encode` domain concepts in existing (textual) general purpose languages?

[www.eclipse.dev/Xtext]



What if…?

… we could easily create languages & tooling specifically for a domain?

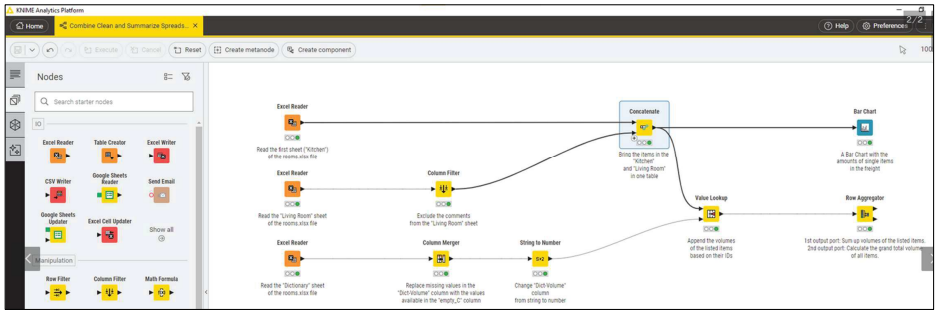
… rather than having to `encode` domain concepts in existing (textual) general purpose languages?

[www.behaviortree.dev]

What if…?

… we could easily create languages & tooling specifically for a domain?

… rather than having to `encode` domain concepts in existing (textual) general purpose languages?



[www.knime.com]

What if…?

… we could easily create languages & tooling specifically for a domain?

… rather than having to `encode` domain concepts in existing (textual) general purpose languages?

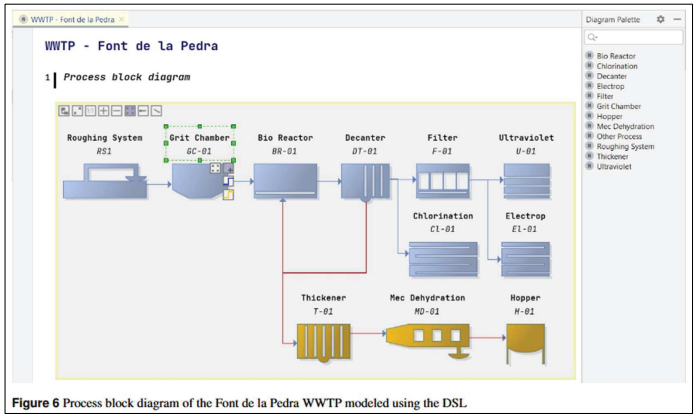


Figure 6 Process block diagram of the Font de la Pedra WWTP modeled using the DSL

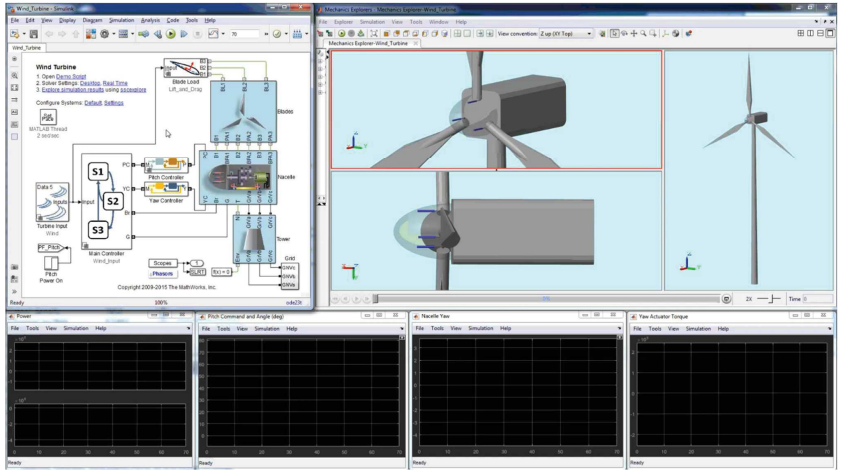
[I. Alfonso et al. A model-based framework for IoT systems in wastewater treatment plants. JOT 22(2). 2023]

What if…?

… we could easily create languages & tooling specifically for a domain?

… rather than having to `encode` domain concepts in existing (textual) general purpose languages?

[www.matlabexpo.com]



Model-driven Software Development

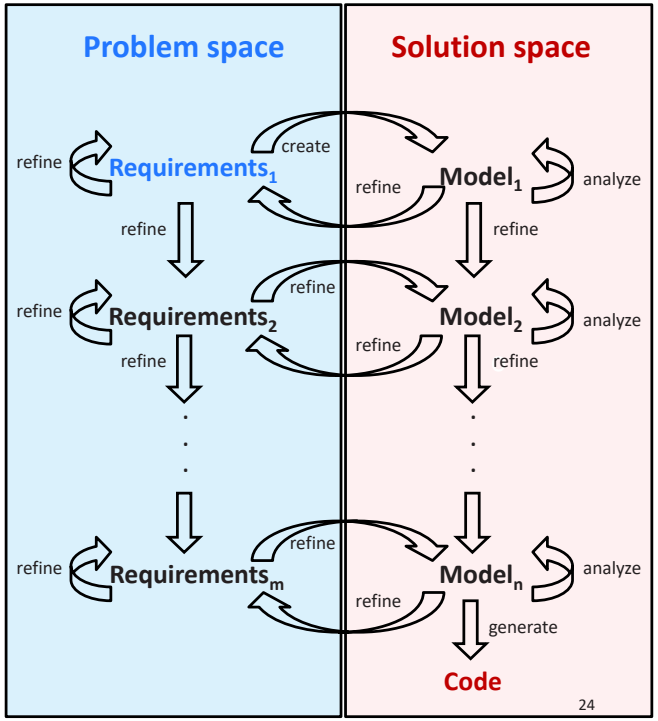
- **Model-driven software development**
  - Models are primary artifact (rather than code)
  - Leveraged in, possibly, all development steps
- **Why?**
  - To make software engineering less 'ad hoc'
  - To achieve
    - reliable designs, and
    - reliable transfer of design intent from specification to implementation
  - To manage complexity by
    - expressing relevant information effectively in the most appropriate format or language (abstraction)
    - automatic creation, transformation, validation (automation)

MDSD Process

Elements in solution space exist in **same medium:** the computer

⇒ Model can gradually evolve into system it is modeling!

⇒ Reduces problems caused by process discontinuities



This Class

- Intro
- Course overview
- Admin stuff

This Course: Content I

- Present some of the
  - key ideas, potential benefits and challenges of **software modeling** in general and
  - of **model-driven development** (MDD) in particular
- Specific attention will be paid to
  - importance of **abstraction** in CS and SW Eng.
  - techniques for the **definition** of modeling languages, and for the **analysis** and transformation of models
  - examples (UML, UML-RT), case studies and tools (RTist, Xtext, Xtend)
- At the end, students will have some critical understanding of
  - **state of the art** in software modeling
  - **theory** and **practice** involving the use, definition, analysis, or transformation of models of software

This Course: Content II

- Balance
  - Lecture and seminar
  - Old (>50 years) and new (<5 years)
  - Theory and practice

- Learn how to summarize & critique papers
- Improve your communication skills

Question 2b: Queen's contributed to: Speaking skills.

Percent

	2005	2006	2007	2008	2009	2010	2011	2012	5 Yr Ave
Applied Science	50	59	57	53	59	57	60	53	57
Arts and Science	57	55	59	57	57	61	54	48	55
Concurrent Education	53	47	55	65	63	66	54	55	61
Education	61	52	56	53	56	53	52	54	54
Law School	71	78	70	74	67	72	81	67	70
School of Business	89	96	90	94	92	87	90	95	92
School of Nursing	56	66	59	58	63	57	43	56	56
Grand Total	59	58	61	59	60	62	57	53	58

[Undergraduate Exit Poll. Queen's University. 2015]

This Course: Structure

- Lectures
  - Containing tool demos
  - Slides will be provided
  - Some have required readings
    - Meant to support, augment lecture content
    - Everybody is expected to have read readings
    - Each will have 1 or 2 'discussion leaders'
    - 20-30mins discussion of reading at beginning of class time
- Assignments
  - 3 assignments on MDSD w/ HCL Model RealTime (aka, RTist, RSARTE, RoseRT)
    - Mac w/ 'Apple silicon' chip anyone?
  - 1 assignment on DSLs with Xtext

This Course: Structure (Cont'd)

- **Project**
  - in groups of 1-2 students
  - I will provide suggestions
  - deliverables
    - project proposal (due around Week 7)
    - presentation (Week 13)
    - final report (due after presentation)

This Course: Evaluation

- **Assignments (4):** **40%**
- **Participation:** **10%**
- **Paper reviews:** **10%**
- **Project, presentation, and report:** **40%**

This Course: Topics

1. **What is a model?**
   - Definitions, key concepts, examples
2. **Models in software engineering**
   - Observations, examples, purposes, characteristics, MDD
3. **Languages**
   - UML, UML-RT
4. **MDSD with UML-RT and HCL Model RealTime**
   - Modeling structure and behaviour w/ UML-RT
   - Testing, code generation, time
   - Assignment 1, 2, and 3
5. **Domain specific languages (DSLs)**
   - Eclipse Modeling Framework (EMF)
   - Abstract & concrete syntax, grammars, meta modeling, model validation, code generation
   - Tools: Xtext, Language: Xtend
   - Assignment 4

This Course: Expected Background

- **Programming**
  - object-oriented
  - experience with Java and Eclipse helpful

This Course: Material

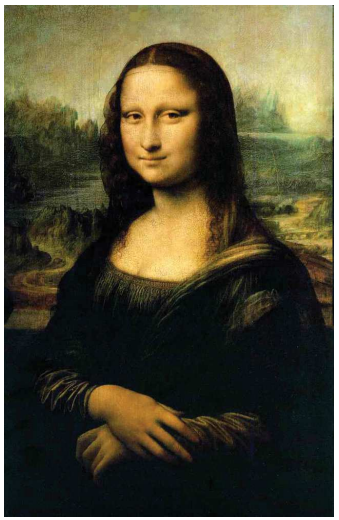
- **Lecture slides**
  - Will be posted
- **Relevant websites:**
  - Course: **www.cs.queensu.ca/~dingel/cisc844_W25**

important

- **Papers:**
  - all online
  - be sure to access publisher's sites from Queen's account

Warning:
Course under Constant Development!

- Want:



- But may end up with: