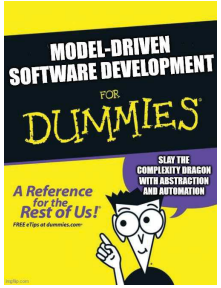
Beyond Code: An Introduction to Model-Driven Software Development (CISC 844)



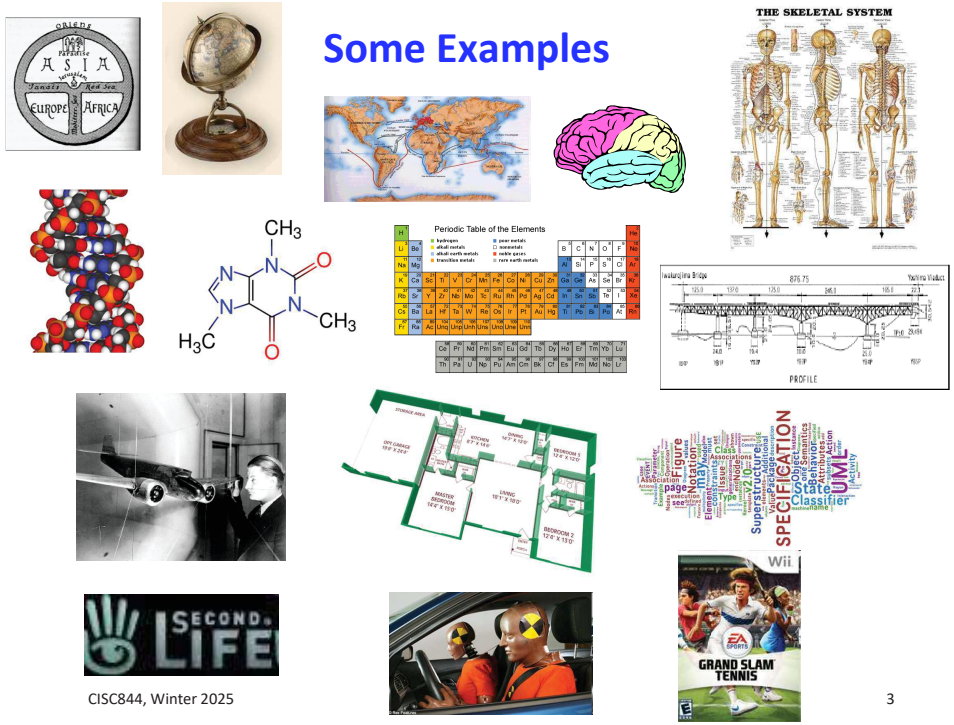
Topic 1: What is a model?

Juergen Dingel
Jan 2025

Today's Class

- **Topic 1: Observations about modeling in general**
  - Examples
  - What is a model?
  - What are they used for?
  - Activities on models
- **Topic 2: Intro to software modeling**
  - What is being modeled?
  - Why?
  - How? I.e., what is a good model?
  - Models as primary artifact
    - Model-Driven Development (MDD)
    - Examples: EGGG, HCL Model RealTime (Rtist), Xtext

Some Examples



Definitions (Cont'd)

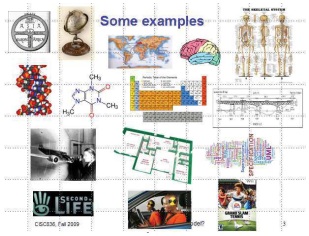
"Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose.

It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose.

A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality.

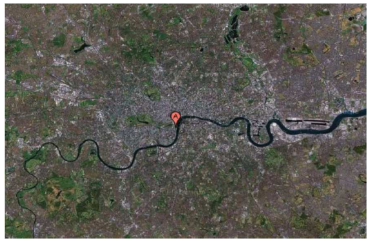
This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality."

Which of these are models in the above sense?



[J. Rothenberg. The Nature of Modeling. In Artificial Intelligence, Simulation, and Modeling, L.E. William, K.A. Loparo, N.R. Nelson, eds. New York, John Wiley and Sons, Inc., 1989, pp. 75-92]

Models, Abstraction, and Interpretation



Interpretation

Add detail

i : Model x Reality

Not necessarily a function

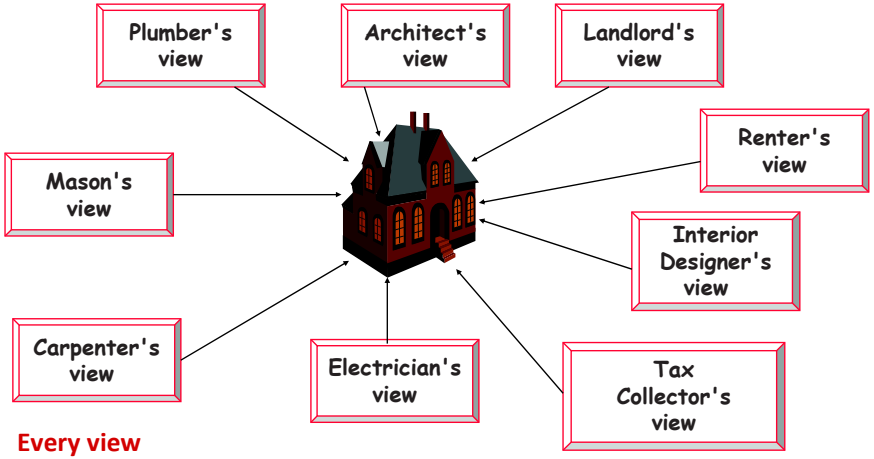
Abstraction

Remove detail

a : Reality → Model

Typically, there's more than one useful abstraction

Models as Views



Every view

- obtained via different abstraction
- may be expressed in different notation (modeling language)
- reflects different intent

[Example from J. Bezivin]

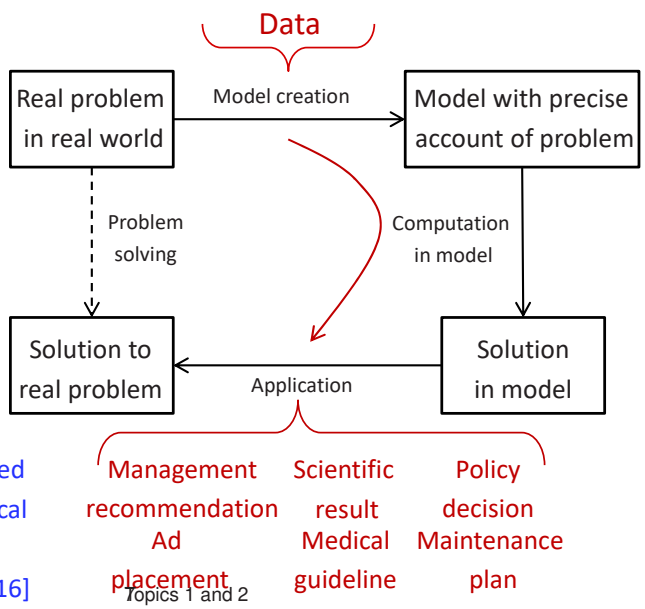
Modeling to Solve Problems

"Executable model of gene expression" [Fisher et al, CAV'16]

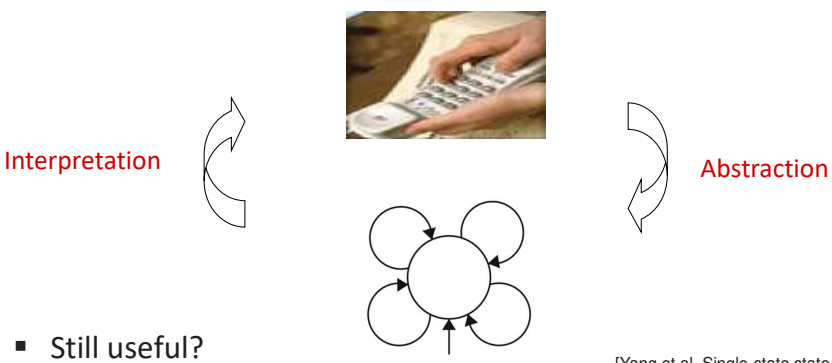
"Russian Track and Field Team Barred From Rio Olympics" [NY Times, 06/17/16]

"CDC Concludes Zika Causes Microcephaly" [04/13/16]

"Programs can be analyzed productively with statistical models" [Devanbu et al, CACM'16]



But, Models Can be Useless, Harmful



- Still useful?
  - Depends on
- Must be careful not to remove too much detail

"Make everything as simple as possible, but no simpler" [A. Einstein]

[Yang et al. Single-state state machines in model-driven software engineering: an exploratory study. EMSE:26:124. 2021]

Models Can be Useless, Harmful (Cont'd)

- Models may only be used for their purpose
- **Example 1:**

« Could I travel from Paris to Anchorage without using a boat? »

« What is the temperature at the bottom if I dig a 100 km deep hole at the surface of the earth ? »

[Example from J. Bezivin]

⇒ Purpose of model must be clear!

Models Can be Useless, Harmful (Cont'd)

If the model is incorrect, the action may be inappropriate

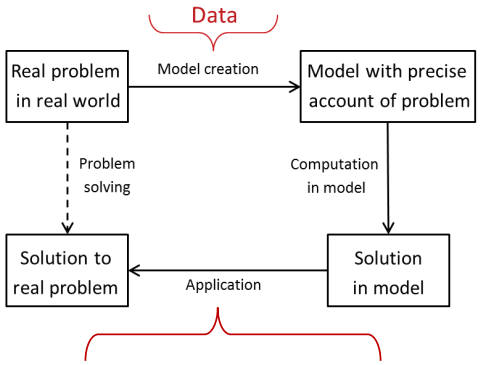
Example 2: The medical technique of bloodletting was based on an incorrect model of the body:

- Hippocrates and many others believed that
  - four crucial elements earth, air, water and fire were balanced within the human body as the four humors: blood, phlegm, and black and yellow bile.
  - disease was due to an imbalance in the four humors.
  - treatment involved restoring their balance through bloodletting.
- In 1799, George Washington died after heavy blood loss sustained in a bloodletting treatment for laryngitis.

[Example from J. Bezivin]

⇒ Model must be validated with respect to purpose

But Models can be Useless, Harmful (Cont'd)



Growth in a Time of Debt
By Carmen M. Reinhart and Kenneth S. Rogoff
American Economic Review: Papers & Proceedings 100 (May 2010): 573–578

https://en.wikipedia.org/wiki/Growth_in_a_Time_of_Debt

Methodology flawed! [2013]

But What Exactly is the Purpose of Models?

On the first glance

- Facilitate/enable analysis, prediction, understanding, decision making
- depending on intent

But need to make important distinction

- **Descriptive models**: model as description

  Reality ⇒ Model ⇒ Predictions & actions

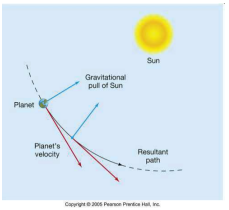
  used in natural sciences and "*backwards engineering*"
- **Prescriptive models**: model as specification

  Reality ⇒ Model ⇒ Actions ⇒ Product

  used in "*forwards engineering*"

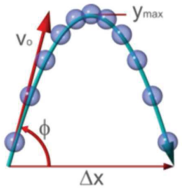
Models in Natural Sciences

- Mostly descriptive
- Facilitate predictions
- Validation via experiments
- Examples:
  - Kepler's laws of planetary motion (ca. 1605)
  - Newton's laws of motion (1687)
    - together with law of gravitation explains Kepler's laws
    - *Law 2:*
      "the acceleration of an object is proportional to the force applied, and inversely proportional to the mass of the object": F = m*a

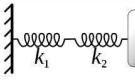
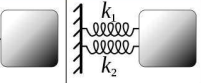


Models in Natural Sciences (Cont'd)

- Examples:
  - Classical mechanics
    - mathematically describe movement of bodies under the influence of forces
    - refinements:
      - quantum mechanics, relativistic mechanics



  - Physics: Coil springs
    - How do springs behave? In isolation? In combination?
    - Hooke's law:
      "the extension of a spring is in direct proportion with the load added to it as long as this load does not exceed the elastic limit": F = -k*x
      where F is force, x is displacement, and k is spring constant

Comparison	In Series	In Parallel
		
Equivalent spring constant	$\frac{1}{k_{eq}} = \frac{1}{k_1} + \frac{1}{k_2}$	$k_{eq} = k_1 + k_2$
Compressed distance	$\frac{x_1}{x_2} = \frac{k_2}{k_1}$	$x_1 = x_2$
Energy stored	$\frac{E_1}{E_2} = \frac{k_2}{k_1}$	$\frac{E_1}{E_2} = \frac{k_1}{k_2}$

Models in Economics

- Examples:
  - Financial models
    - MONIAC (Monetary National Income Analogue Computer) to model the national economic processes of the UK (1942)
      http://en.wikipedia.org/wiki/Moniac

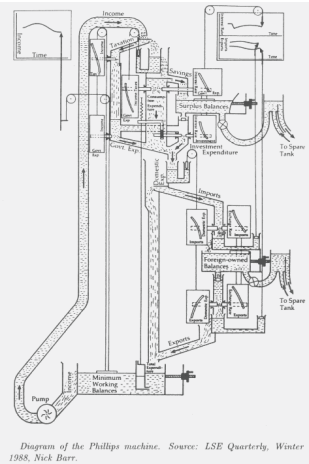
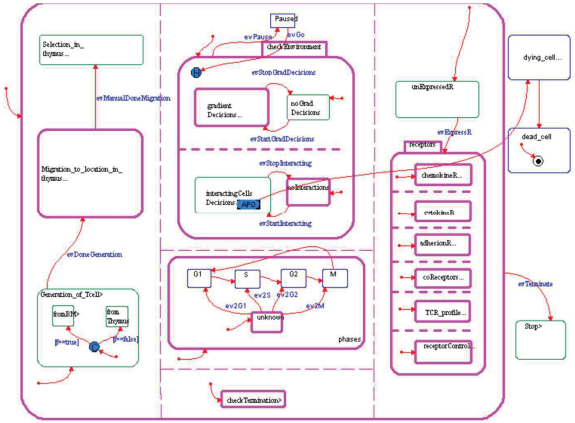


Diagram of the Phillips machine. Source: LSE Quarterly, Winter 1988, Nick Barr.

  - http://www.nytimes.com/interactive/2015/12/16/upshot/fed-interest-rates-rube-goldberg-machine.html

Models in Biology

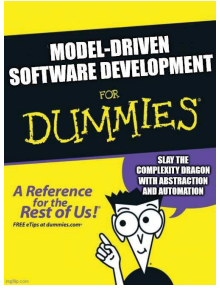
- Examples:
  - Model of T cell development in the thymus gland [EHC03]



Models in Entertainment

- Appeal comes from both
  - being like reality, and
  - not being like reality

CISC 844: Models in Software Development: Methods, Techniques and Tools



Topic 2: Models in Software Development

Juergen Dingel
Jan 2025

Overview

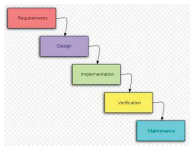
- What is being modeled?
- Why?
- How? I.e., what is a good model?
- Models as primary artifact
  - Model-Driven development (MDD)
  - Examples
    - EGGG
    - IBM RSARTE, IBM Rhapsody, HCL Rtist, Model RealTime
    - Unity
    - Xtext
    - EMF

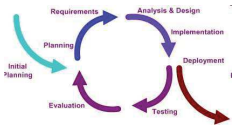
Observations

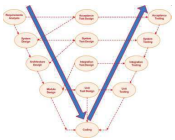
- Models in software engineering (SE) can be both **prescriptive** (e.g., requirements models) or **descriptive**:
  - e.g., models extracted from source code for program understanding, reverse engineering, impact analysis, test case generation, etc
- Creation, adoption, analysis of models
  - in SE not as well-understood as in other engineering disciplines
- MDD (and also this course) focuses more on prescriptive models
  - Following courses are more about automatic creation and use of descriptive models of code
    - ELEC 875: Design Recovery and Automated Evolution (Tom Dean)
    - ELEC 876: Software Reengineering (Jenny Zou)

What is being Modeled?

- Almost any artifact created/used during development:
  - Models of structure
    - data, execution state (object graphs), of languages (e.g., grammars and meta models), variability models
  - Models of software behaviour
    - on different levels: requirements, design, architecture, code
    - different parts: individual operations, methods, interactions (b/w processes, user and interface, or system and environment)
    - faults, performance, processes, workflows, environment, test models, variability models (for software product lines)
  - Models of software development processes





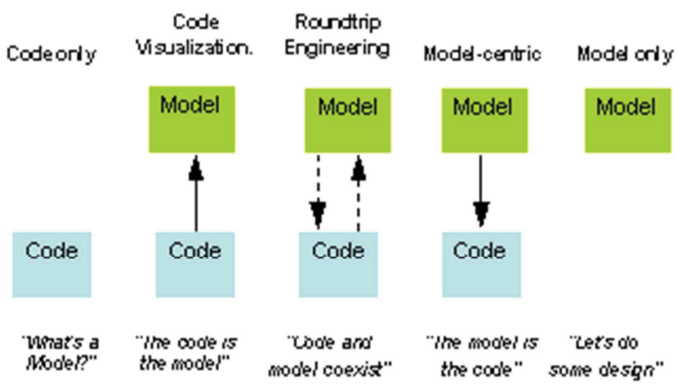


Purpose: Dealing with Complexity

- *"build models, because cannot comprehend system in its entirety"* [BRJ05, p6-7]
- **Essential complexity** [GS04,p36]
  - "inherent to problem being solved", "number of features, relationships, and dependencies that must be considered when decomposing problem"
  - cannot be reduced or eliminated
  - **Example**: online ordering process handled by single company vs online ordering process handled by multiple companies
- **Accidental complexity** [GS04,p36]
  - "an artifact of solution", "number of features, relationships, and dependencies that must be considered when composing solution"
  - can be reduced
  - **Examples**:
    - online commerce application written in assembly language vs one written in Java, C# etc
    - platform (e.g., J2EE, .NET, Corba) complexity [Sch06]

Purpose (Cont'd)

- Documentation (e.g., for design decisions), visualization, specification, design templates, understanding/analysis, round-tripping, code generation

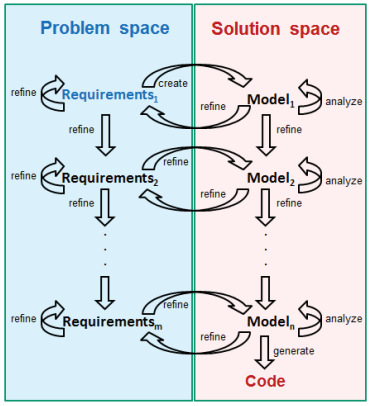


Purpose (Cont'd)

- Execution and analysis
  - **checking static (i.e., structural) properties**
    - e.g., control flow graphs, ER diagrams, DB schemas, Alloy [Jac02]
  - **checking dynamic (i.e., behavioural) properties**
    - e.g., model checking [BA10, Eme08]
      - component/process interaction [KMR02, MC01]
      - navigation in web applications [HEBB10]
    - performance analysis [Pet09, HLT+04]
  - **test case generation**
    - for embedded (automotive) systems [SYR08]
    - incrementally for software product lines [UGKB08]
  - **"what if" analyses [Sch06]**

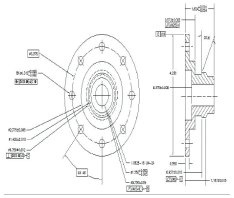
Model-Driven Development (MDD)

- **Core ideas [Sel03][KSLB03][Sch06]**
  - Abstraction and automation [Sel03]
    - Automation, e.g., for model transformation, code generation, analysis
    - performance of current code generators within 5-15% of manually implemented systems
  - Iterative development and successive refinement via automated model transformations
  - Analysis



Model-Driven Development (MDD) (Cont'd)

- **The potential power of these ideas**
  - **MDSD = CAD/CAM for software**



  - **[Sel03]: software engineering medium best positioned to benefit from modeling, b/c**
    - model can gradually evolve into final product w/o the risks that "discontinuities", e.g., in material, bring
    - conducive to incremental iterative development, b/c no discontinuities that preclude backtracking

How well does MDSD currently realize this vision?

Who is Practising MBSE?

IBM Rational

Sampling of Embedded Software Developed Using MDD

Automated doors, Base Station, Billing (In Telephone Switches), Broadband Access, Gateway, Camera, Car Audio, Convertible roof controller, Control Systems, DSL, Elevators, Embedded Control, GPS, Engine Monitoring, Entertainment, Fault Management, Military Data/Voice Communications, Missile Systems, Executable Architecture (Simulation), DNA Sequencing, Industrial Laser Control, Karaoke, Media Gateway, Modeling Of Software Architectures, Medical Devices, Military And Aerospace, Mobile Phone (GSM/3G), Modem, Automated Concrete Mixing Factory, Private Branch Exchange (PBX), Operations And Maintenance, Optical Switching, Industrial Robot, Phone, Radio Network Controller, Routing, Operational Logic, Security and fire monitoring systems, Surgical Robot, Surveillance Systems, Testing And Instrumentation Equipment, Train Control, Train to Signal box Communications, Voice Over IP, Wafer Processing, Wireless Phone

16 ON DEMAND BUSINESS

Model-Driven Software Development (MDSD): Examples

- **Industrial: E.g.,**
  - Motorola [BLW05], OCE [DS02], EADS (Scade), [ADSAA04], GM, avionics, automotive
- **Academic: E.g.,**
  - EGGG [Orw00]
  - Feature-oriented programming [BD07,BSR04,BG97]
  - Software Factories (MS) [GS04]
  - Language-oriented programming [MPS09, LOP09]
  - Language workbench [Fow09]
- **Supporting commercial tools: E.g.,**
  - HCL DevOps Model RealTime (formerly, HCL Rtist)
  - IBM Rational: RSARTE (RoseRT), Rhapsody
  - The MathWorks: MATLAB StateFlow/Simulink
  - MetaCase: MetaEdit
  - No Magic: Cameo Systems Modeler
  - MS Visual Studio [GS04]

MDSD Example: EGGG

- **EGGG: Extensible Graphical Game Generator**
- Jon Orwant's PhD 1999
- Exploit commonalities among games to write software that automatically generates the code for a game

High-level description of game G → EGGG → Software implementing G

- **Key questions:**
  - Which games can be supported? How to best describe a game? How write the code generator?
- Domain analysis results in identification of concepts that allow description of certain class of games in terms of models represented in a domain-specific language (DSL)

MDSD Example: EGGG (Cont'd)

```
game is poker

turns alternate clockwise

Discard means player removes 0..3 cards or 4 cards if Ace
Fold means player loses

2..6 players

game is Shuffle(deck) and Deal(cards, 5) and (bet(money) or Fold)
        and Discard(hand, N) and Deal(cards, 5-N)
        and (bet(money) or Fold) and compare(cards)

StraightFlush is (R, S) and (R-1, S) and (R-2, S) and (R-3, S) and (R-4, S)
FourKind is (R, s) and (R, s) and (R, s) and (R, s)
FullHouse is (R, s) and (R, s) and (R, s) and (Q, s) and (Q, s)
Flush is (r, S) and (r, S) and (r, S) and (r, S) and (r, S)
Straight is (R, s) and (R-1, s) and (R-2, s) and (R-3, s) and (R-4, s)
ThreeKind is (R, s) and (R, s) and (R, s)
TwoPair is (R, s) and (R, s) and (Q, s) and (Q, s)
Pair is (R, s) and (R, s)
HighCard is (R, s)

hands are [StraightFlush, FourKind, FullHouse, Flush, Straight,
           ThreeKind, TwoPair, Pair, HighCard]

hand is five cards
goal is highest(hand)
```

Domain-specific language (DSL)

software that writes software!

EGGG

MDSD Example: EGGG (Cont'd)

- **What makes this work?**
  - Thorough domain analysis
    - identify communalities & differences, relevant concepts, parameters & their ranges, rules & principles, patterns & anti-patterns, classification, scope of application
      ⇒ develop suitable meta model
    - develop code generator
    - develop user-friendly way allowing users to describe an element within the scope (i.e., a model)
      ⇒ find suitable concrete syntax

abstraction

automation

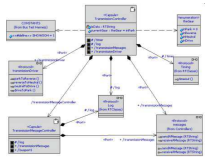
- **Consequences**
  - Deep understanding required
  - Higher upfront cost, but once in place, development much simplified

> *"We need to know what we are doing before we can automate it. A DSM solution is implausible when building an application or a feature unlike anything developed earlier".*
>
> *"Early adopters of domain-specific modeling have been enjoying productivity increases of 500-1000% in production for over 10 years now"*
>
> [KT08, p18]

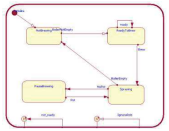
MDSD Example: HCL Model RealTime

- HCL Model RealTime

capsule diagram (structure)



state machine diagram (one per capsule) (behaviour)



abstraction

HCL RTist

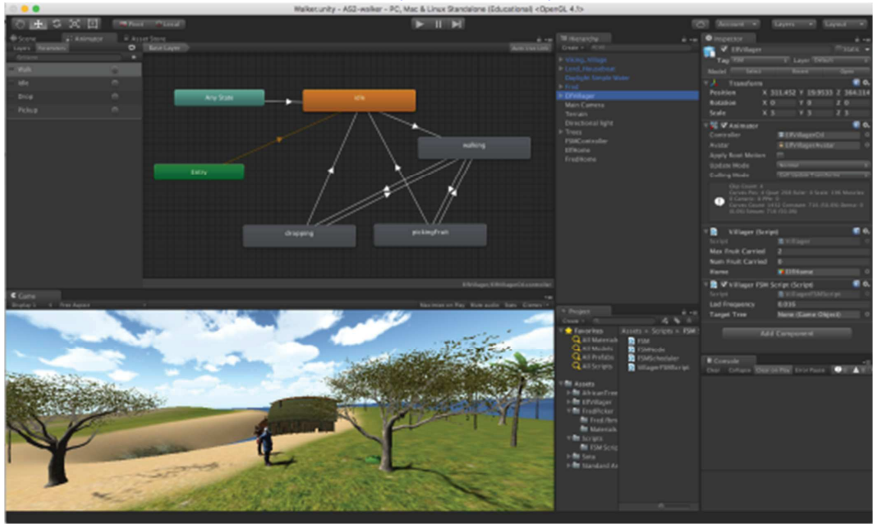
automation

C++

- UML-RT as DSL (UML variant) [Sel98]
  - for timed, reactive systems

MDSD Example: Game Industry

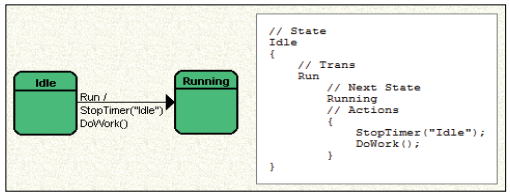
http://docs.unity3d.com/Manual/Animator.html

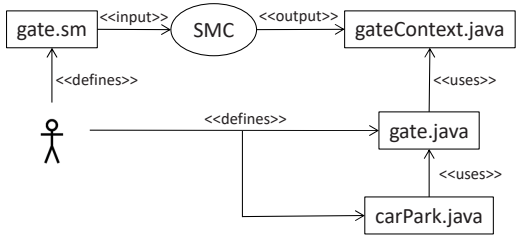


Screenshot courtesy Nick Graham

SMC: The State Machine Compiler

- SMC allows
  - textual definition of state machine
  - automatic generation of code implementing that state machine
  - visualization of state machine
- Supports 15 languages
- Generated code uses State Design Pattern





http://smc.sourceforge.net/

XText

Xtext

- Eclipse-based open-source framework for development of programming languages and domain-specific languages
- Offers
  - Parser generator
  - Editor plugin generator supporting
    - Syntax highlighting
    - Well-formedness checking (validation) w/ error markers and quick fixes
    - Background parsing
    - Auto-completion with content assist
    - Hyperlinking connecting uses with declarations
    - Hovering
    - Folding and outline view
  - Support for
    - Code generation (using Xtend, a variant of Java)
    - Interpretation, translation to Java
  - Large user community, http://www.eclipse.org/Xtext/community.html

"A language is only as good as its supporting tooling"
[B. Selic]

CISC836, Winter 2018 DSLs 61

Next in CISC 844

- Topic 3: Expressing software models
  - UML
  - UML-RT
- Topic 4: MDD with UML-RT
  - HCL Model RealTime
- Topic 5: Using software models
  - Code generation with EMF
- Topic 6: DSLs