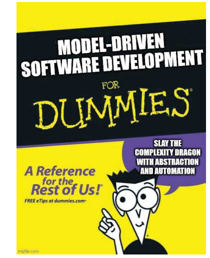
Queen's University | Faculty of ARTS AND SCIENCE

Classroom **PROCEDURES**

- Get vaccinated
- Provide vaccination proof
- Do the daily COVID screen
- Don't attend when ill
- Wear a mask
- Leave room promptly
- Wash hands frequently
- Don't consume drinks/food

QUartsci.com/Fall2021

Beyond Code: An Introduction to Model-Driven Software Development (CISC836)



Topic 0: Intro & Motivation, Overview, Admin

Juergen Dingel
Sept 2021

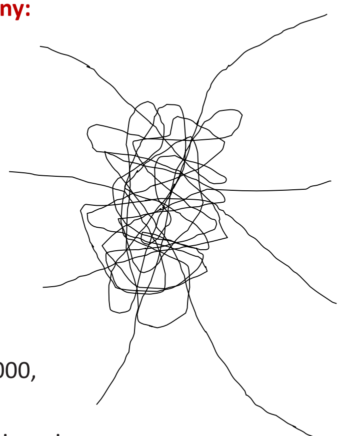
About Me

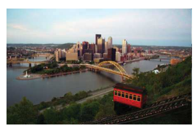
Small town Germany: Born, raised, etc

Berlin: UG

Pittsburgh: PhD

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





A

B

C

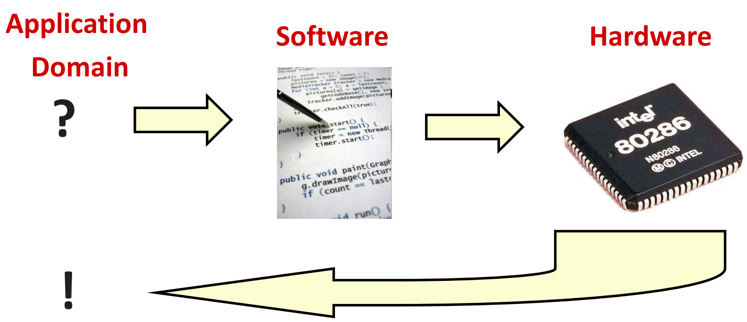
D

This Lecture

- **Motivation**
  - Software development is hard
  - It won't get any easier
  - Need more powerful techniques and tools (things that start with the letter "A")
- **Course overview**
- **Admin stuff**

What is Software?

> *"The programs, routines, and symbolic languages that control the functioning of the hardware and direct its operation."*
>
> American Heritage Dictionary



What is Software: Crucial

- **Crucial to functioning of modern society**
  - critical infrastructure
    - transportation
    - energy
    - water
    - communication
  - business and finance
  - health care
  - military
  - entertainment
  - education
  - …

Software Complexity: In LOC

- **Windows OSs**
  - NT 3.1 (1993): 0.5 million
  - 95:
    - 11 m
    - > 20(
  - 2000: 2
  - XP (2001): 35 million
  - Vista (2007): 50 million
  - Wi
- **Windo**
  - Off
  - Off
  - Vis
- **Mac OS X "Tiger":** 85 million

- **Average iPhone app:** 40,000 LoC
- **Pacemaker:** 100,000 LoC
- oC
- **Boeing 787:** 14 million
- **F-35 fighter jet:** 24 million
- **Large Hadron Collider:** 50 million

1BTkUbdqE

- 2005: 10 million
- 2014: 100 million

Software is one of the most complex man-made artifacts!

But perhaps "Lines of code" is a poor measure of complexity?!

[1 million LoC = 18,000 pages of printed text = stack 6 feet high]

[Charette, Why Software Fails, IEEE Spectrum, Sept 2005]
[McCandless, www.informationisbeautiful.net/visualizations/million-lines-of-code]

Software Complexity: In State Space Size

- **State s of a program P**
  -
  -
- **Sta**
  - set of reachable states of P
- **State spaces can be very large**
  - in … values
- **Corre … achable states**
  - P ⊨ φ often means ∀ s∈reachable(P). s ⊨ φ

What is the size of the state space of the software in your car?

Software is one of the most complex man-made artifacts!

Consequences of this Complexity (Cont'd)

- **Failing software**
  - **money**
    - **Examples**: ESA Ariane 5, Mars Climate Orbiter, Skype bug in '07, blackout in '04, MS Zune bug in '09, US telephone system, …
    - Cost of SW errors in US in 2001:

      US$ 60 billion [US National Inst. of Standards & Technology]
    - Worldwide cost of IT failure (est.):

      US$ 3000 billion [ZDNet12]
    - High IT project failure rates:

      51 (24%) of 214 IT projects cancelled [BCS08]

[ZDNet12] http://www.zdnet.com/article/worldwide-cost-of-it-failure-revisited-3-trillion/

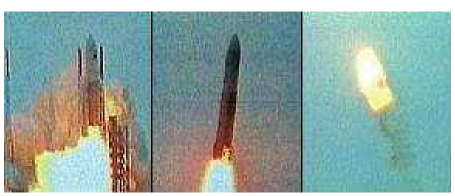
[BCS08] McManus, Wood-Harper. A study in project failure. British Computer Society CS. 2008

Consequences of this Complexity (Cont'd)

- **Failing software**
  - **money**
  - **lives**
    - Therac 25, …
- **More details**
  - Peter Neumann's **http://www.risks.org**
  - Ivars Peterson. Fatal Defect: Chasing Killer Computer Bugs. Vintage Books, New York, 1996.

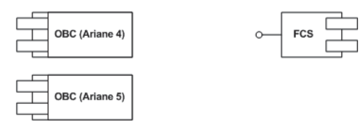
Example: ESA Ariane 5 (June 1996)

- On June 4, 1996, unmanned Ariane 5 launched by ESA explodes 40 seconds after lift-off
- One decade of development costing $7billion lost
- Rocket and cargo valued at $500million destroyed



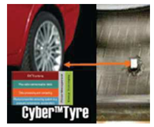
- What went wrong?
  - Bad reuse of code from Ariane 4
  - Bad fault-tolerance mechanism
  - Bad coding practices

Example: ESA Ariane 5 (June 1996) (Cont'd)



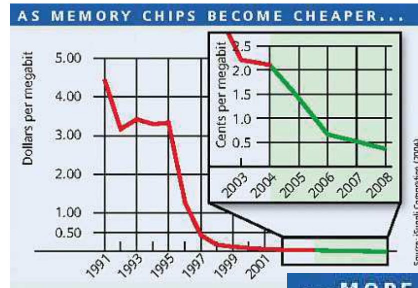
- Example of how not to do reuse:
  - Parts of Flight Control System (FCS) taken from Ariane 4
  - Horizontal velocity much greater for Ariane 5
  - Unprotected conversion operation in FCS causes error
  - On-board computer (OBC) interprets error code as flight data
  - …
  - Launcher self-destructs
- Example of how not to achieve fault-tolerance:
  - FCS and backup FCS identical, thus backup also failed
- Example of how not to code:
  - When code caused exception, it wasn't even needed anymore
- References:
  - [Gle96] and **www.ima.umn.edu/~arnold/disasters/ariane.html**

Trends

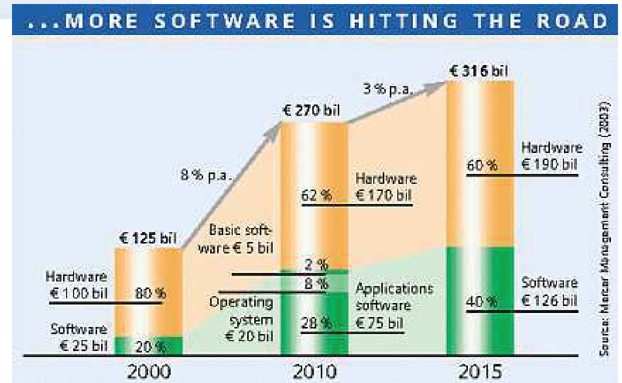


- **More complexity**
  - Less mechanical; more electronic & computerized
  - More features & capabilities
  - Less stand-alone; more integration, distribution and concurrency
  - Increasing virtualization ("software-defined" everything)
  - M. Andreesen. "Why software is eating the world". WSJ, Aug 20, 2011.
- **More regulatory oversight**

Example: Automotive Software



1991: $4.5/Mbit
2008: $0.004/Mbit
=> less than 1/1000



Automotive Software

...

Safety

Security

More integration

More functionality

"Excited" vs "Not Prepared"

"An exciting new era of change is sweeping the global automotive industry.
In fact, I believe the industry will experience more change in the next 5 years than in it has in the last 50 years."

GM CEO Mary Barra
March 12, 2015

"Only 19% of [175] interviewed auto executives describe their organizations as prepared for challenges on the way to 2025.
Just 33% said their organizations are adaptable to face challenges."

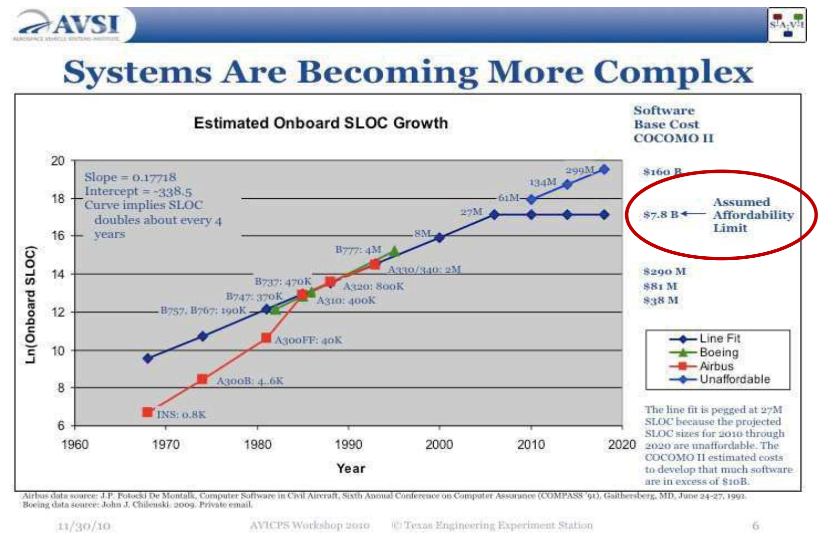
IBM

Stanley, Gyimesi.
Automotive 2025: Industry w/o borders.
January 2015

From Workshop on Modeling in Automotive SE

Robert Baillargeon, MASE'15, Sept 27, 2015

But Automotive is No Exception



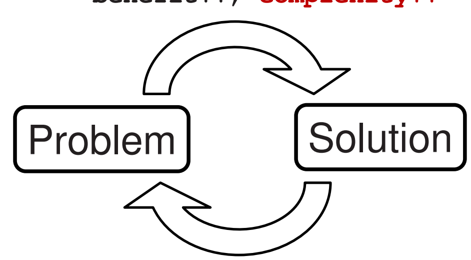
Examples: Systems of Systems

- Government
  - IRS tax system: 100 million lines of code
- Health care
  - HL7 standards (**www.hl7.org**)
    - for exchange, management and integration of electronic healthcare information
- Energy
  - "smart-grid" projects in US
- Transportation
- Business and finance
- Military
- Communications

A Global, Societal Phenomenon?

Complexity as problem solving strategy [Tai96]

`benefit++; complexity++`



[Tai96] J.A. Tainter. Complexity, problem solving, and sustainable societies. In R. Costanza, O. Segura, and J. Martinez-Alier, editors, Getting Down to Earth: Practical Applications of Ecological Economics. Island Press, 1996.

Growing Dependency

- *"We're surrounded by systems that, if they fail, can injure people or ruin them economically. Examples include automobile control systems, banking software, telecommunication software, and just about any industrial control software"*

Stroustrup: Software Development for Infrastructure. Computer 5(1), 2012

- *"… [the] cyber threat is one of the most serious economic and national security challenges we face as a nation"*

US President Barack Obama, May 29, 2009

The Challenge

Capabilities	↑
Size	↑
Complexity	↑
Costs	?
Failures	?

Ariane 5 explosion, June 4, 1996

What Can We Do?

…

Safety

Security

More integration

More functionality

Weapons to tame complexity

- abstraction
- automation
- analysis
- decomposition
- reuse

key ingredients to MDE (and engineering in general)

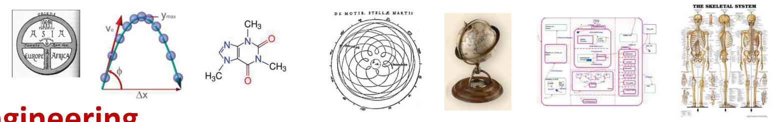
This is Not a New Idea

- Modeling in other disciplines
- Abstraction in the history of computing

Modeling in other Disciplines

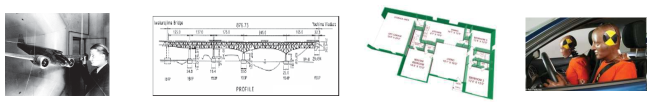
Natural sciences

Understanding, predicting existing phenomena ("Backwards Engineering")



Engineering

Building artifacts with certain properties ("Forwards Engineering")



Entertainment

Doing what normally would be impossible

Modeling is central, except in SW Eng

Modeling in other Disciplines (Cont'd)

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)

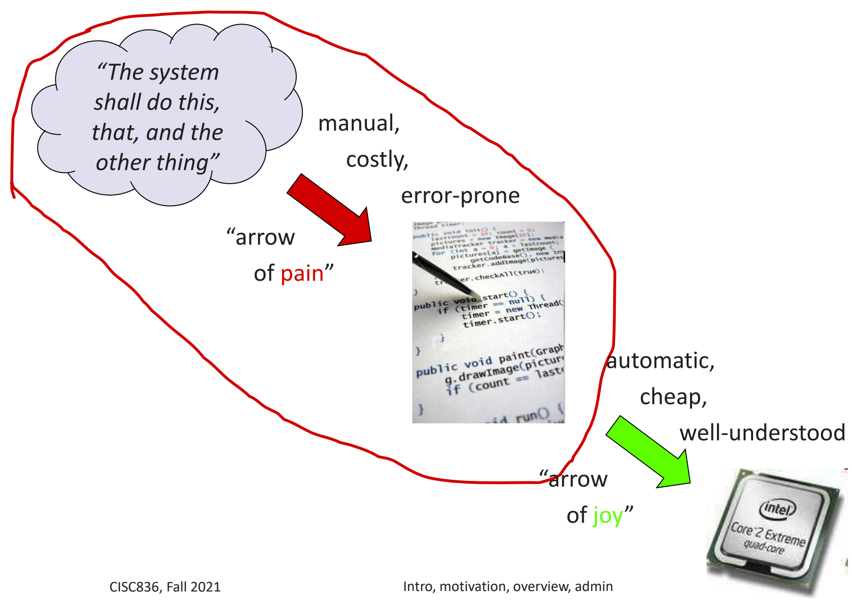
Software Engineering still has a long way to go…

Abstraction & the History of 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"

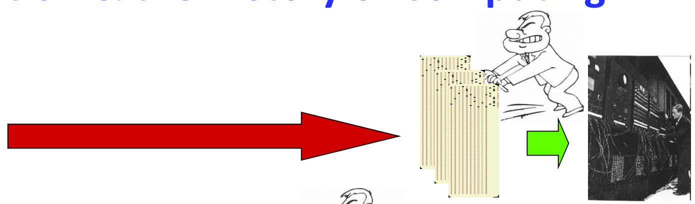
Software Development from 30,000 Feet



Abstraction & the History of Computing

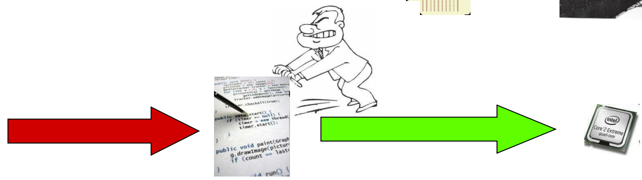
In the past

"The system shall do this, that, and the other thing"

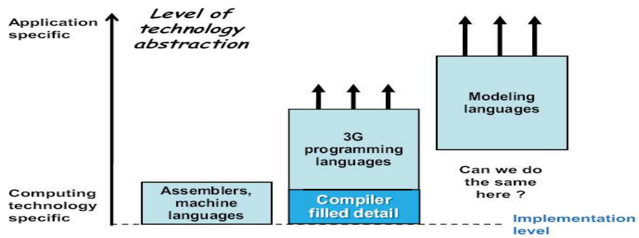


Today

"The system shall do this, that, and the other thing"



Or, more technically



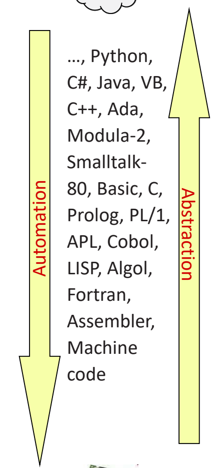
[B. Selic]

Abstraction & the History of 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

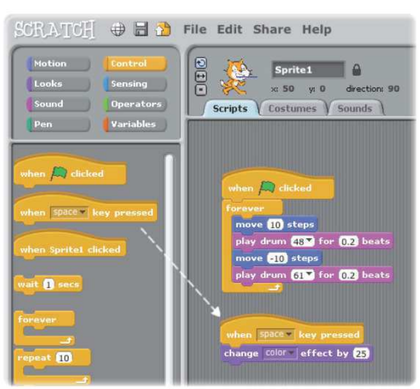
but what makes this work in practice is

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

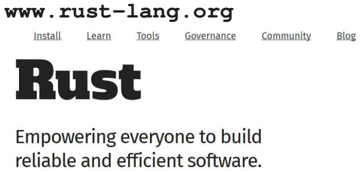


Note that We Are Talking About More Than "Better Programming Languages"

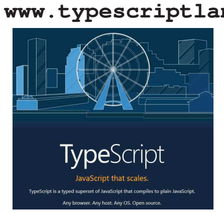
scratch.mit.edu



www.rust-lang.org



www.typescriptlang.org



golang.org

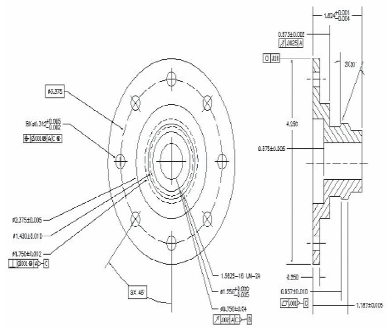
So, **abstraction** and **automation** are a good team, but let's see what can happen if we throw **analysis** into the mix as well…

MDD 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

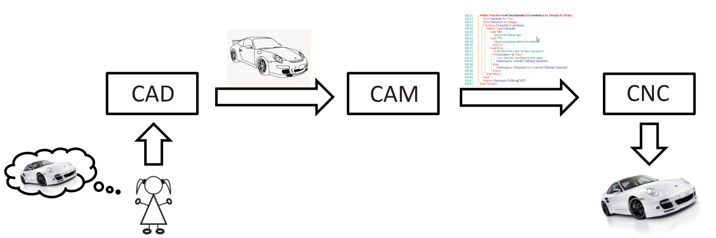


MDD in Manufacturing (Cont'd)

- **Example: Concorde (1976 – 2003)**
  - > 100,000 drawings
  - in 2 languages, using both metric and imperial systems
  - ⇒ worked, but 7x over budget

MDD 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**

Model-Driven Development (MDD)

Improve productivity, quality, and ability to handle complexity by

- increasing level of **abstraction**
  - through use of models
- leveraging **automation**
  - e.g., via code generation from models
- improving **analysis** capabilities
  - e.g., through executable models

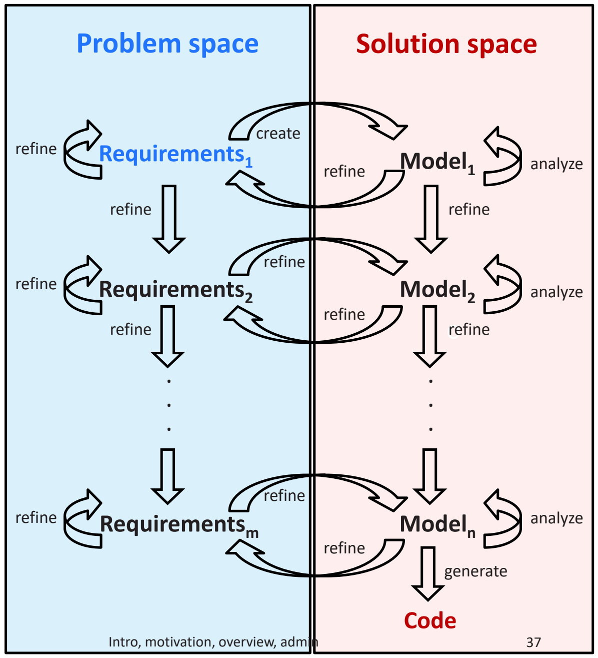
MDD = Abstraction + Automation + Analysis

MDD 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 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 (Papyrus, 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	57	70
School of Business	89	96	90	94	92	87	90	95	92
School of Nursing	56	66	59	58	63	57	43	58	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/ IBM RSARTE
  - 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 Papyrus-RT**
   - 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

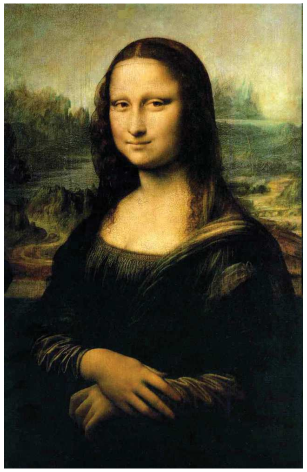
- **Programming**
  - object-oriented
  - experience with Java and Eclipse helpful

This Course: Material

- **Lecture slides**
  - Will be posted
- **Relevant websites:**
  - Course: **www.cs.queensu.ca/~dingel/cisc836_F21** 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: