

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

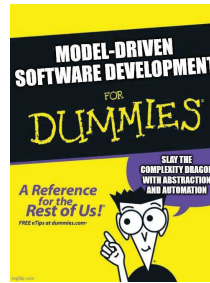
Topic 0: Intro & Motivation, Overview, Admin

Juergen Dingel
Jan 2026

CISC844, Winter 2026

Intro, motivation, overview, admin

1



You?

About Me

Small town Germany:

Born, raised, etc

Berlin: UG

Pittsburgh: PhD

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

CISC844, Winter 2026

Intro, motivation,
overview, admin



A



B



C



D

2

CISC844, Winter 2026

Intro, motivation,
overview, admin

3

CISC844, Winter 2026

Intro, motivation, overview, admin

4



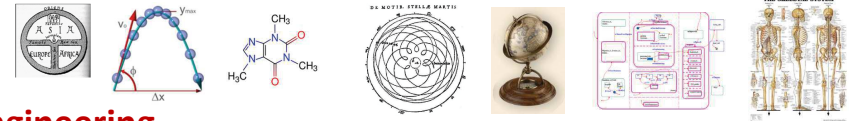
Modeling

- Models as purposeful abstractions
- Not a new idea...

Modeling in other Disciplines

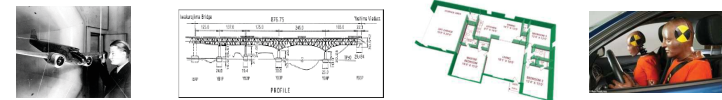
Natural sciences

Understanding, predicting existing phenomena (“Backwards Engineering”)



Engineering

Building artifacts with certain properties (“Forwards Engineering”)



Entertainment

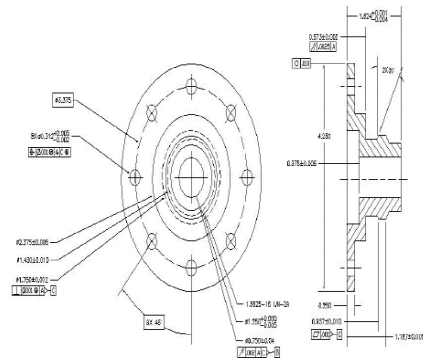
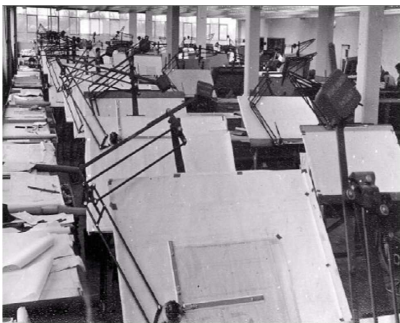
Doing what normally would be impossible



Modeling in Manufacturing

Mechanical design from 1800 to about 1980:

- Draftsmen create 3-view drawings
 - Machinists create parts from drawings
- ⇒ laborious, error-prone, inefficient



Modeling in Manufacturing (Cont'd)

Example: Concorde (1976 – 2003)

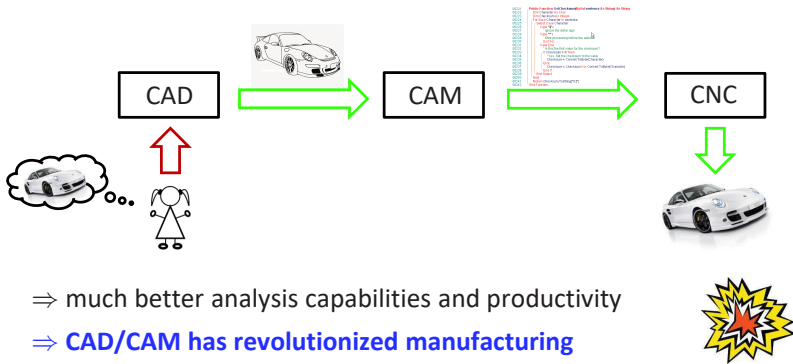
- > 100,000 drawings
 - in 2 languages, using both metric and imperial systems
- ⇒ worked, but 7x over budget



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)



CISC844, Winter 2026

Intro, motivation, overview, admin

9

**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"

CISC844, Winter 2026

Intro, motivation, overview, admin

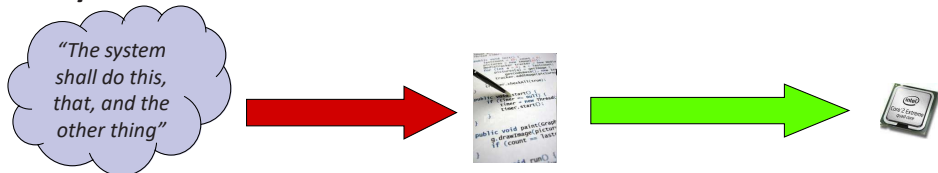
10

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

Early computing



Today



CISC844, Winter 2026

Intro, motivation, overview, admin

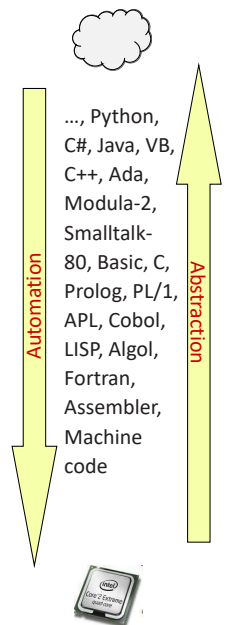
11

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



CISC844, Winter 2026

Intro, motivation, overview, admin

12

... 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

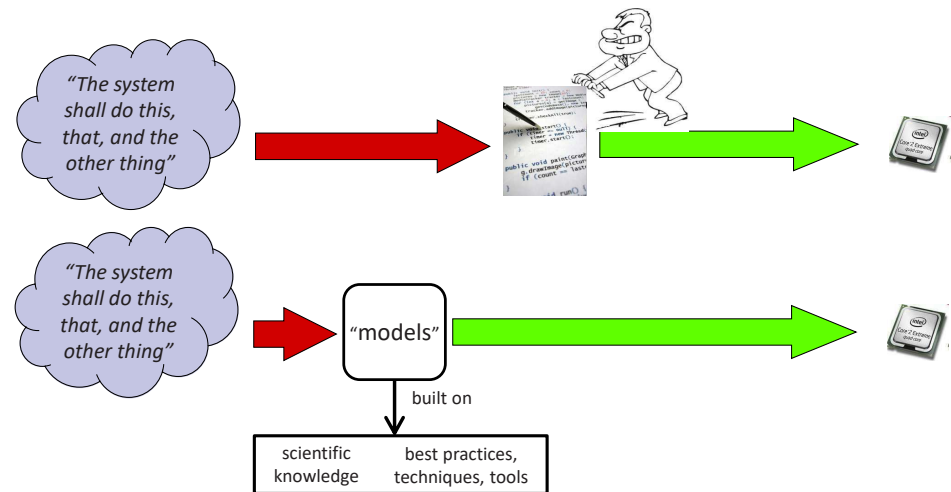
Characteristics

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

Making Software Engineering considerably more 'ad-hoc'!

What if... ?

... we made models the **primary artifact** in software engineering?

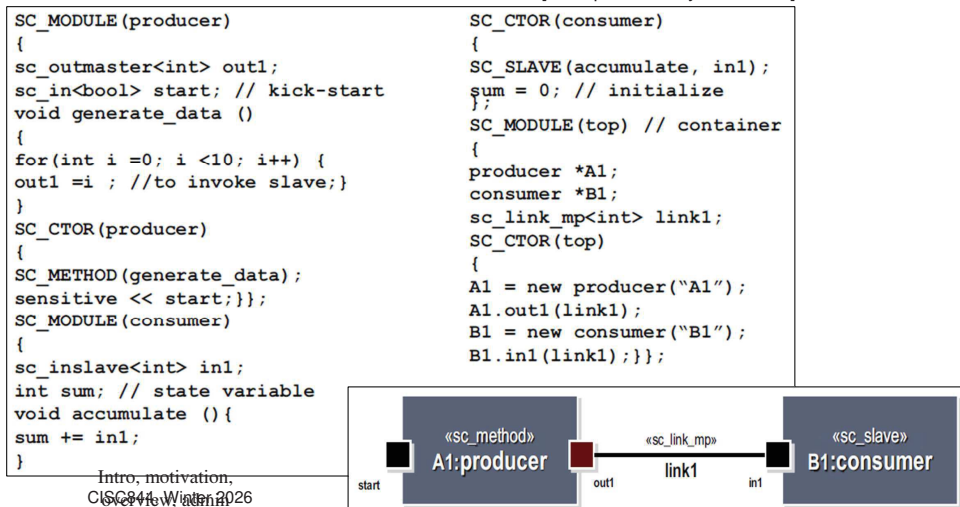


What if...?

... the **architecture of the system was not hidden** in the code?

... **design models** were part of development?

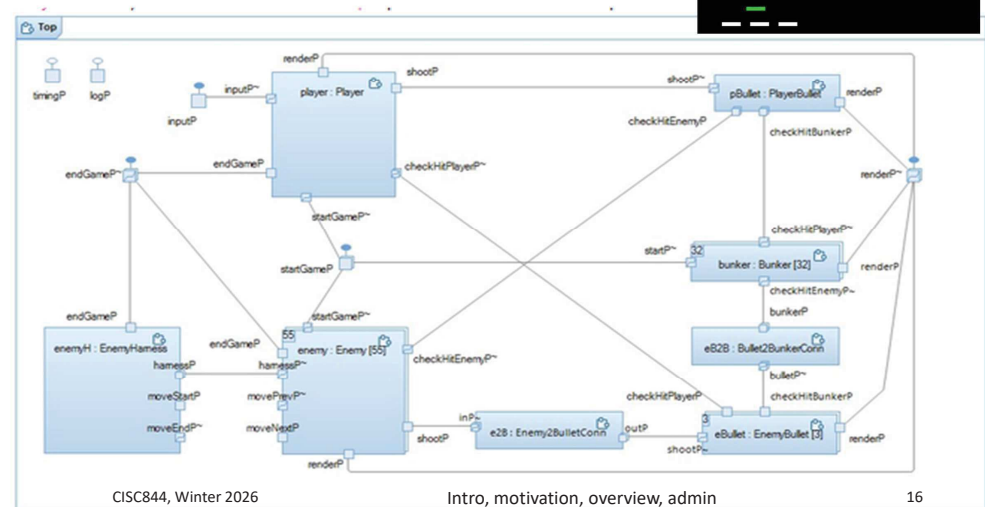
[example courtesy Bran Selic]



What if...?

... the **architecture of the system was not hidden** in code?

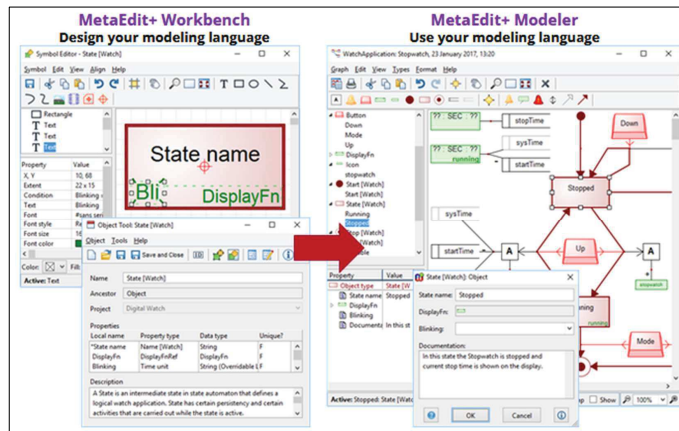
... **design models** were part of development?



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]



CISC844, Winter 2026

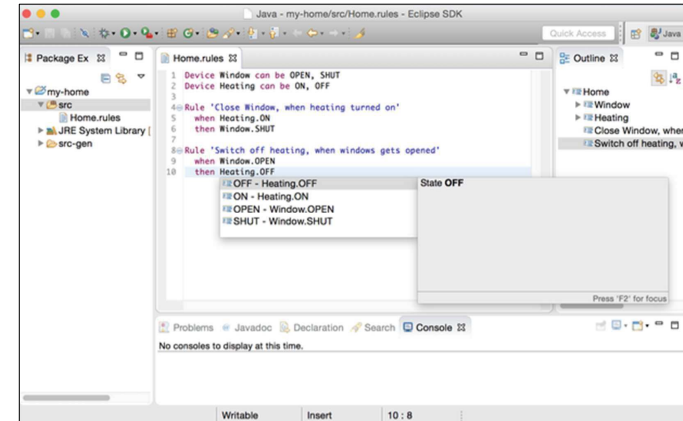
Intro, motivation, overview, admin

17

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]



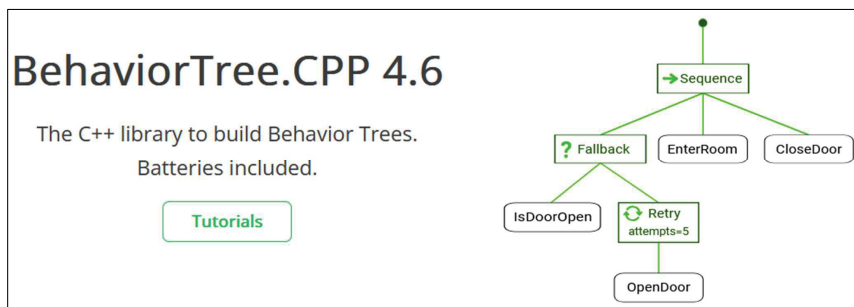
CISC844, Winter 2026

Intro, motivation, overview, admin

18

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]

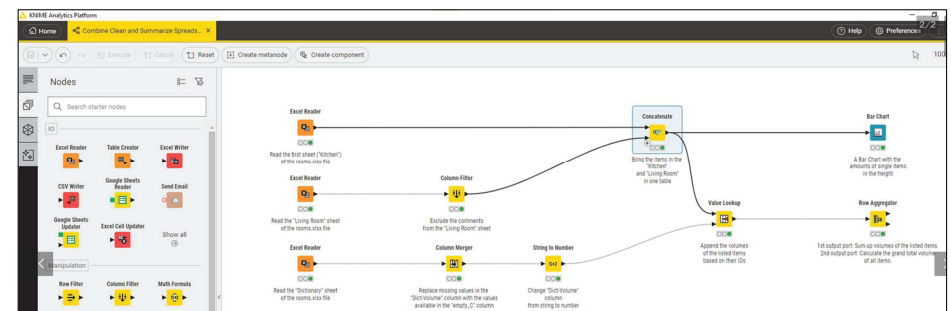
CISC844, Winter 2026

Intro, motivation, overview, admin

19

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]

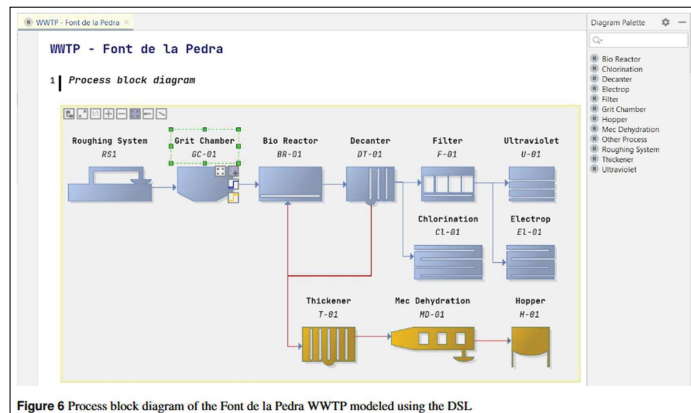
CISC844, Winter 2026

Intro, motivation, overview, admin

20

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?

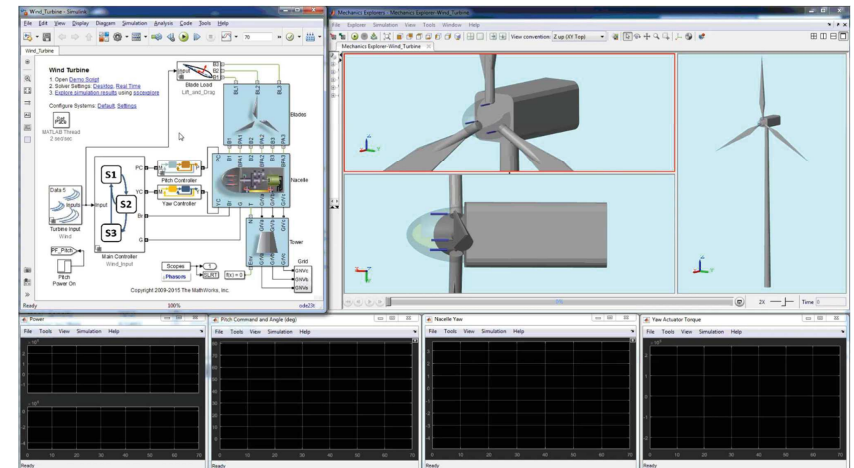


[I. Alfonso et al.
 A model-based
 framework for IoT
 systems in
 wastewater
 treatment plants.
 JOT 22(2). 2023]

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]



CISC844, Winter 2026

Intro, motivation, overview, admin

21

What if...?

... we could easily **generate code from natural language prompts?**

“Vibe coding encourages **shallow understanding**. It prioritizes what looks right now over what will hold up six months from now. **Architecture decisions get made implicitly** by the model. Patterns get embedded without review. And before long, you’re swimming in **generated complexity that no one quite understands** --- not even the person who shipped it.”

“You’re no longer just writing features. You’re **defining the environment in which those features get created**. That means **owning the architecture, enforcing patterns, and designing systems** that guide not just human teammates, but increasingly AI collaborators”

C. Adam. “Agile is Out, Architecture is Back”. Medium. Sept 2025.
medium.com/@craig_32726/agile-is-out-architecture-is-back-7586910ab810

CISC844, Winter 2026

Intro, motivation, overview, admin

23

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 **manage complexity** by
 - expressing relevant information effectively in the most appropriate format or language (**abstraction**)
 - automatic creation, transformation, validation (**automation**)
 - To achieve
 - **reliable designs** and
 - **reliable transfer of design intent from specification to implementation**

CISC844, Winter 2026

Intro, motivation, overview, admin

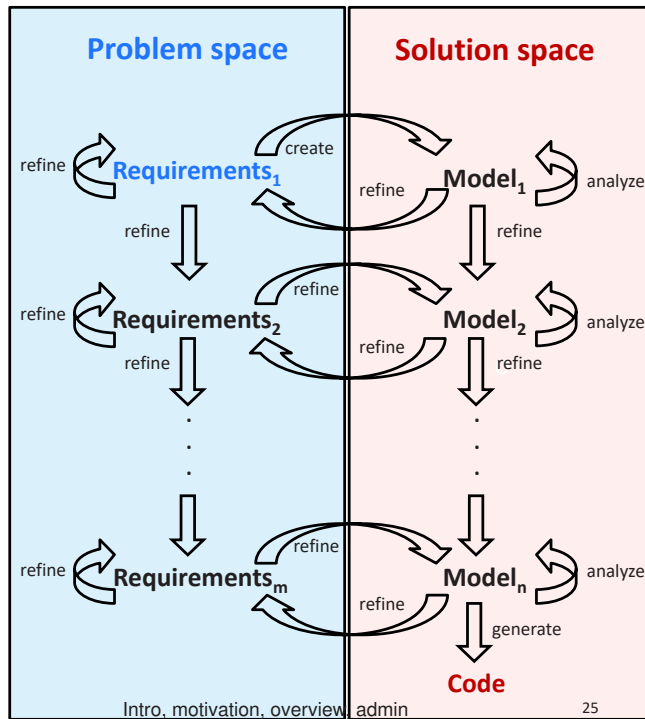
24

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**

CISC844, Winter 2026

CISC844, Winter 2026

Intro, motivation, overview, admin

26

This Course: Content I

- **Present some of the**
 - key ideas, potential benefits and challenges of **software modeling** in general and
 - of **model-driven development** 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 (HCL Model RT, HCL Code RT, Eclipse Xtext, Eclipse 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	58	56
Grand Total	59	58	61	59	60	62	57	53	58

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

CISC844, Winter 2026

Intro, motivation, overview, admin

27

CISC844, Winter 2026

Intro, motivation, overview, admin

28

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/Code RealTime
 - 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

- Modeling structure and behaviour of communicating state-based systems
- Graphically: Assignments 1 and 2 (HCL Model RealTime)
- Textually & graphically: Assignment 3 (HCL Code RealTime)

5. Domain specific languages (DSLs)

- Eclipse Modeling Framework (EMF)
- Abstract & concrete syntax, grammars, meta modeling, 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_W26

important

■ Papers

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

■ OnQ

- for assignment submission only

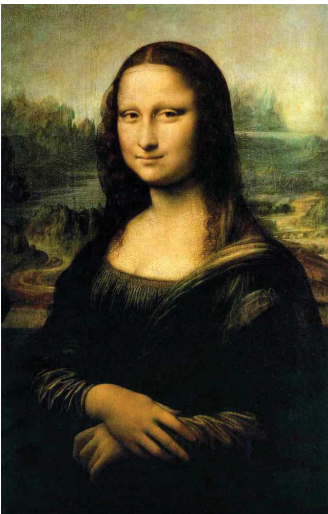
■ Discourse discussion forum

- questions about anything

Warning:

Course under Constant Development!

■ Want:



■ But may end up with:

