Prospective Students
Undergrad Students
Grad Students
News and Events
Seminars
Staff
Faculty
Alumni
About Us
Contacts
Research
Internal Users
Innovation Council

Distinguished Seminar Series

Image Understanding & Web Security

Henry S. Baird
Palo Alto Research Center, CA, USA
Thursday, December 4, 2003

Robot-Assisted Image-Guided Needle Placement Prostate Interventions: Specific application to Prostate Cancers

Gabor Fichtinger
Johns Hopkins University
Thursday, December 11, 2003

Models, Software Models, and Model-Driven Engineering

Branislav Selic
IBM Software Group - Rational Software
Ottawa, Ontario, Canada
Thursday, February 12, 2004

Slides from his Seminar

Challenges for Bioinformatics Systems

Gregory Butler
Department of Computer Science
Concordia University
Thursday, February 26, 2004

Database Issues in Stream Data Management

Tamer Ozsu
Department of Computer Science
University of Waterloo, Canada
Thursday, May 6, 2004

Multipoint Communication in Optical Networks

Ahmed E. Kamal
Department of Electrical & Comp. Eng.
Iowa State University, Ames, Iowa
Thursday, June 3, 2004
2:30 p.m.   BIO 1102

The **Distinguished Seminar Series** is sponsored by the School of Computing and

CITO
Communications and Information Technology Ontario

Better Connections.
Better Technology.
Better Results.

Queen's University

School of Computing

Distinguished Seminar Series

Image Understanding & Web Security

Henry S. Baird

Palo Alto Research Center
Palo Alto, CA 94304

Internet services offered for human use are suffering abuse by programs ('bots, spiders, scrapers, spammers, etc). We mount a defense against such attacks with CAPTCHAs, `completely automatic public Turing tests to tell computers and humans apart.' These are special cases of `human interactive proofs' (HIPs), a class of security protocols allowing people easily to identify themselves over networks as members of given groups. I will review the five years of evolution of HIP R&D, highlights of the first NSF HIP workshop, and applications of HIPs now in use and on the horizon.

One of the best ways to build a CAPTCHA is to exploit the gap in ability between humans and machines in attempting to read images of text. I will describe two such reading-based CAPTCHAs, developed in collaborations between PARC and UC Berkeley:

PessimalPrint, motivated by studies of physics-based image degradations, uses images synthesized pseudo-randomly over certain words, typefaces, and image qualities; and

BaffleText, motivated by the psychophysics of human reading, uses image-masking degradations that seem to require Gestalt perception skills that people are especially good at.

Both of these CAPTCHAs have been validated by experiments on human subjects and commercial OCR machines, and both have successfully resisted attack by advanced computer-vision techniques (so far) . I'll offer proposals for an image understanding research agenda to advance further the state of the art of web security.

[Joint work with Richard Fateman, Allison Coates, Kris Popat, Monica Chew, Tom Breuel, & Mark Luk.]

Dr. Baird is a Principal Scientist and manager of the Statistical Pattern & Image Analysis research area at the Palo Alto Research Center, a subsidiary of Xerox. He has published three books and sixty-eight technical articles, and holds seven patents. He has taught at Princeton and UC Berkeley, and is a Fellow of the IEEE and of the IAPR. With Manuel Blum of CMU, he organized the 1st NSF Int'l HIP Workshop, held at PARC in January 2002.

Thursday December 4, 2003.
2:30-3:30 PM.
Walter Light WLH 205

The **Distinguished Seminar Series** is sponsored by the School of Computing and

CITO
Communications and Information Technology Ontario

Better Connections. Better Technology. Better Results.

Send comments about these Web pages to webmaster@cs.queensu.ca

Queen's UNIVERSITY

School of Computing

Distinguished Seminar Series

Robot-Assisted Image-Guided Needle Placement Prostate Interventions: Specific application to Prostate Cancers

Gabor Fichtinger

Johns Hopkins University,
Baltimore, MD

Image-guided needle-based interventions, like brachytherapy, biopsy, and tissue ablations represent a fast growing segment of our healthcare today. Surgical robotics can assist execute these procedures with enhanced accuracy, reliably, and consistency in routine setup, in a manner that is affordable outside the confines of excellent research hospitals. Robot-assisted surgical systems are rather complex: in addition to all issues of general robotics, they also involve the problems pertaining to surgical visualization, modeling, planning, execution, monitoring, and system-level integration. The overall goal of the talk is to give an overview of our ongoing program and present specific approaches to many of the aforementioned technological problems. Solutions for the management of prostate cancer will also be highlighted.

Biography

Gabor Fichtinger received BS and MS in Electrical Engineering, and PhD Computer Science from the Technical University of Budapest, Hungary, in 1986, 1988, and 1990, respectively. He has been in the U.S. since 1990. During the first few years of his career, Dr. Fichtinger was involved in computer graphics and biomedical visualization. Later his work progressed to architecting stereotactic radiosurgery and computer-assisted neurosurgical planning systems for actual clinical use, in both academic and industry settings. For the last couple of years, his focus has been the research of computer-integrated surgery that includes the problems of surgical visualization, modeling, planning, execution, monitoring, and system-level integration of all those. His specialty is robot-assisted image-guided needle-placement procedures, primarily for cancer therapy. Since 1999, Dr. Fichtinger has been at the Johns Hopkins University, as Director of Engineering of the Center for Computer-Integrated Surgery and Associate Research Professor of Computer Science and Radiology.

Thursday December 11, 2003.
2:30-3:30 PM.
Walter Light WLH 205

The **Distinguished Seminar Series** is sponsored by the School of Computing and

CITO
Communications and Information Technology Ontario

Better Connections.
Better Technology.
Better Results.

School of Computing

Distinguished Seminar Series

Models, Software Models, and Model-Driven Engineering

Branislav Selic

IBM Software Group - Rational Software
Ottawa, Ontario, Canada

Models have played a crucial role in the development of complex engineering systems from ancient times to the present day. However, the use of models in software design is not only relatively rare, but is actively rejected by many practitioners. This is the case despite the fact that many software systems are among the most complex engineering systems ever devised and where one might expect that the potential for abstraction that models provide would be essential. In this talk, we first explain the reasons for this apparent paradox and then describe the essential concepts behind the newly-emerging "model-driven" approach to software design and development. In particular, we focus on UML 2, which is representative of the next generation of modeling languages designed specifically for model-driven development. We demonstrate how this language can be used to specify systems at different levels of abstraction, starting with the highest (architectural) levels, and then explain how such models are automatically transformed into implementations. The talk concludes with a brief review of the state of the art in industrial application of model-driven methods

Biography

Bran Selic is a Distinguished Engineer at IBM Rational Software in Kanata, Canada and an Adjunct Professor of Computer Science at Carleton University in Ottawa. He has over three decades of industrial experience in the design and evelopment of complex software systems in a variety of different domains. He is the primary author of one of the earliest technical textbooks (published in 1994) that describes practical model-driven software development techniques. Bran has been involved with the definition and standardization of the UML language from 1996 and is currently chairing the task force responsible for standardizing the UML 2.0 revision. He has published widely and has given numerous invited talks on model-driven development and related topics.

Thursday, February 12, 2004
2:30-3:30 PM.
Walter Light WLH 210

The **Distinguished Seminar Series** is sponsored by the School of Computing and

CITO
Communications and Information Technology Ontario

Better Connections.
Better Technology.
Better Results.

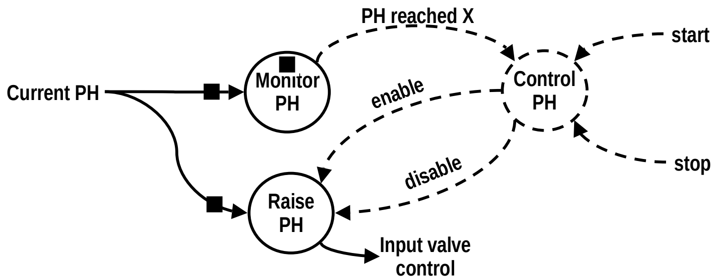
Rational software

Models, Software Models, and Model-Driven Engineering

Bran Selic
Distinguished Engineer
IBM Rational Software – Canada
bselic@ca.ibm.com

- Objective:
  - To clarify what this “model-driven” hype is all about
  - To explain what we can do with it today
- Getting there:
  - On engineering models and software models
  - Model-driven development and MDA
  - UML 2.0 – a language for MDD/MDA

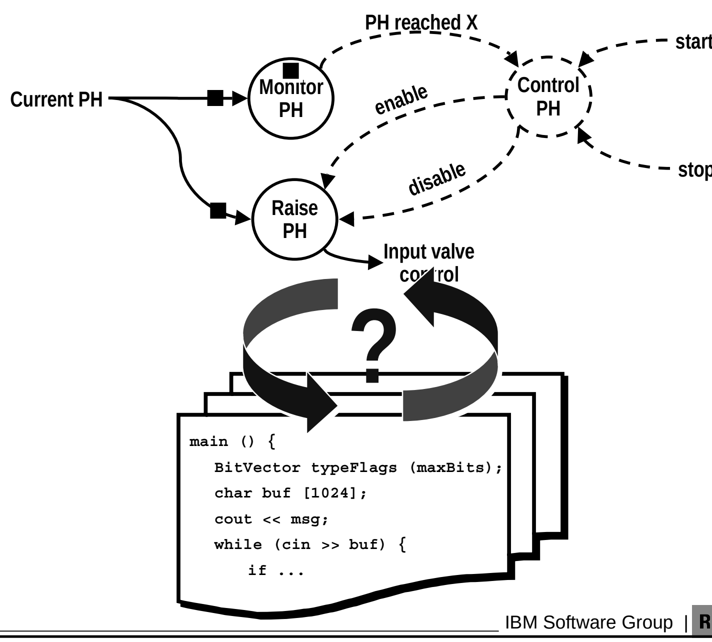
A Skeptic's View of Software Models…



"…bubbles and arrows, as opposed to programs, …never crash"

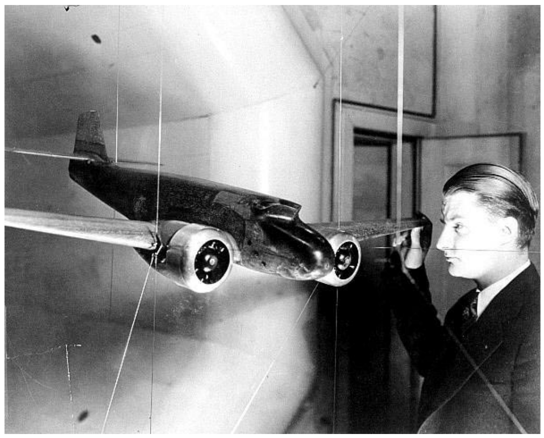
-- B. Meyer
"UML: The Positive Spin"
American Programmer, 1997

The Problem with Bubbles…



Models in Traditional Engineering

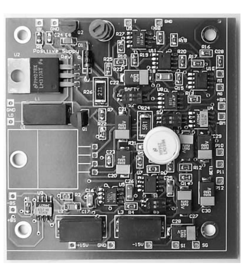
- As old as engineering (e.g., Vitruvius)
- Traditional means of reducing engineering risk



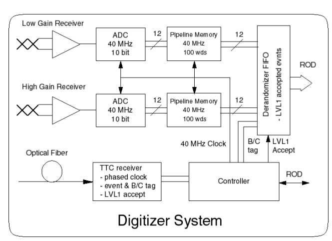
Engineering Models

- Engineering model:

  A reduced representation of some system that highlights the properties of interest from a given viewpoint



Modeled system

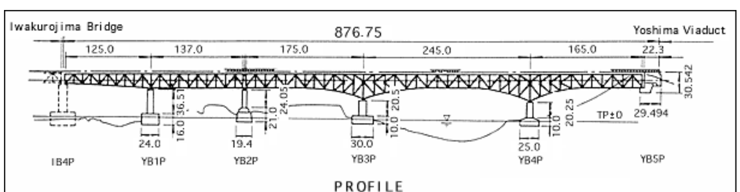


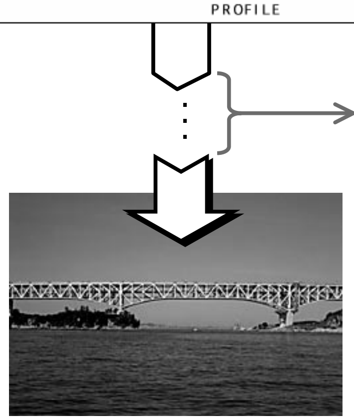
Functional Model

How Engineering Models are Used

1. To help us understand complex systems
   - Useful for both *requirements* and *designs*
   - Minimize risk by detecting errors and omissions early in the design cycle (at low cost)
     - Through analysis and experimentation
     - Investigate and compare alternative solutions
   - To communicate understanding
     - Stakeholders: Clients, users, implementers, testers, documenters, etc.
2. To drive implementation
   - The model as a blueprint for construction

A Common Problem with Engineering Models





Semantic Gap **due to:**

- **Idiosyncrasies of actual construction materials**
- **Construction methods**
- **Scaling effects**
- **Skill sets**
- **Misunderstandings**

Can lead to serious errors and discrepancies in the realization

Characteristics of Useful Models

IBM

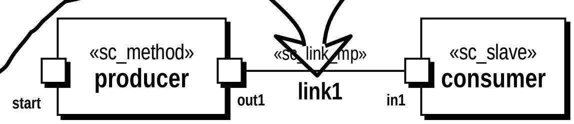
- Abstract
  - Emphasize important aspects while removing irrelevant ones
- Understandable
  - Expressed in a form that is readily understood by observers
- Accurate
  - Faithfully represents the modeled system
- Predictive
  - Can be used to answer questions about the modeled system
- Inexpensive
  - Much cheaper to construct and study than the modeled system

> *To be useful, engineering models must satisfy all of these characteristics!*

Models of Software

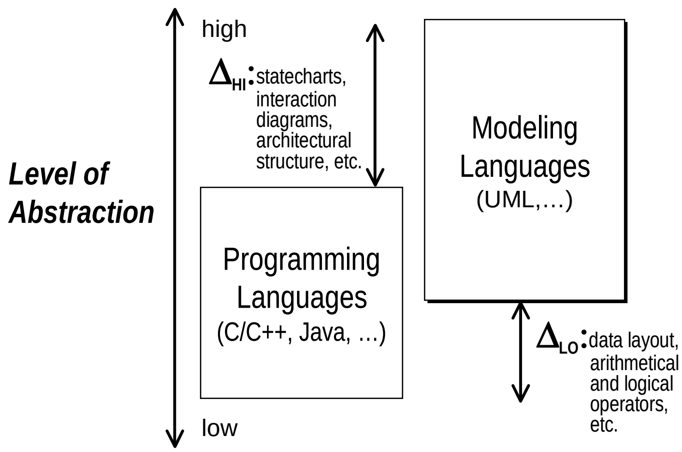
IBM

```
SC_MODULE(producer)
{
sc_outmaster<int> out1;
sc_in<bool> start; // to kick-start the producer
void generate_data ()
{
for(int i =0; i <10; i++).{
out1 =i ; //this will invoke the slave;}
}
SC_CTOR(producer)
{
SC_METHOD(generate_data);
sensitive << start;}};
SC_MODULE(consumer)
{
sc_inslave<int> in1;
int sum; // declare as a module state variable
void accumulate (){
sum += in1;
cout << "Sum = " << sum << endl;}
SC_CTOR(consumer)
{
SC_SLAVE(accumulate, in1);
sum = 0; // initialize the accumulator}};
SC_MODULE(top) // structural module
{
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);}};
```

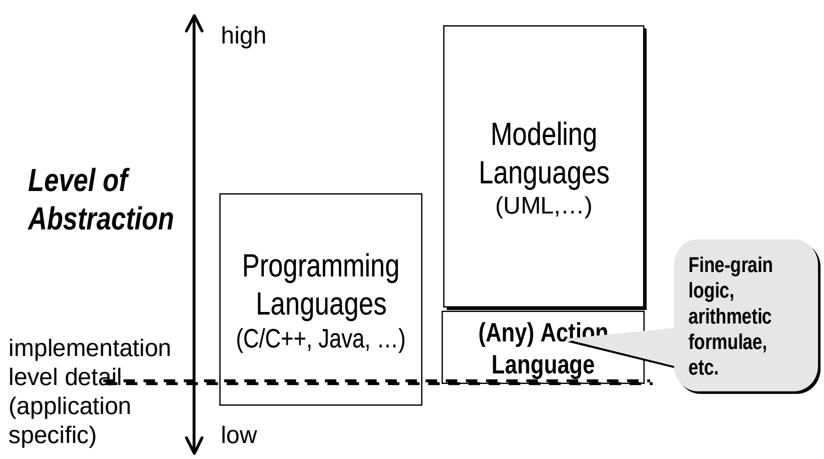


Modeling versus Programming Languages

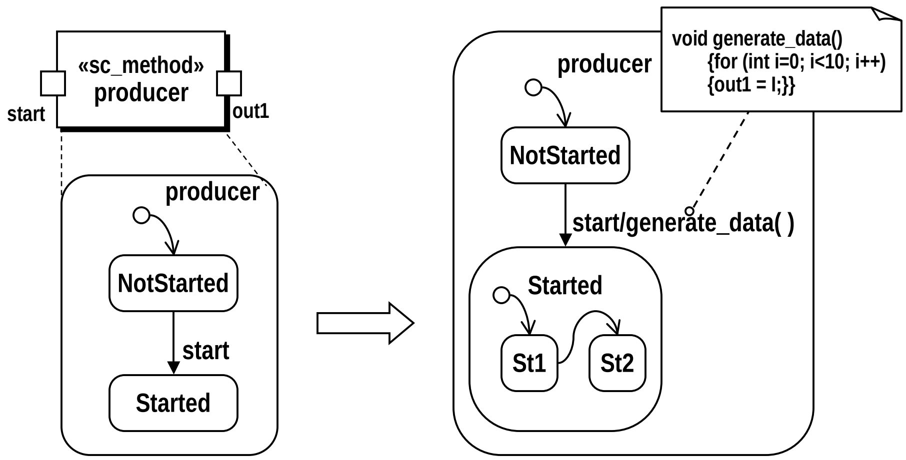
- Cover different ranges of abstraction



Covering the Full Range of Detail



Model Evolution: Refinement



- Detail can be added continuously until the specification is complete

The Remarkable Thing About Software

> *Software has the rare property that it allows us to directly evolve models into full-fledged implementations without changing the engineering medium, tools, or methods!*

⇒ This ensures perfect accuracy of software models; since the model and the system that it models are the same thing

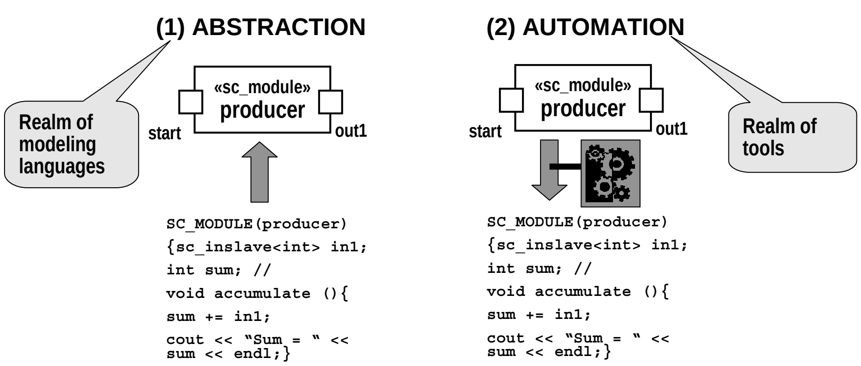
The model evolves into the system it was modeling

Model-Driven Style of Development (MDD)

- An approach to software development in which the focus and primary artifacts of development are models (as opposed to programs)
- Some implications
  - Automatic generation of programs from models
  - Modeling languages as implementation tools

The Essence of MDD

- Based on two proven complementary techniques:



Primary Forms of Automation for MDD

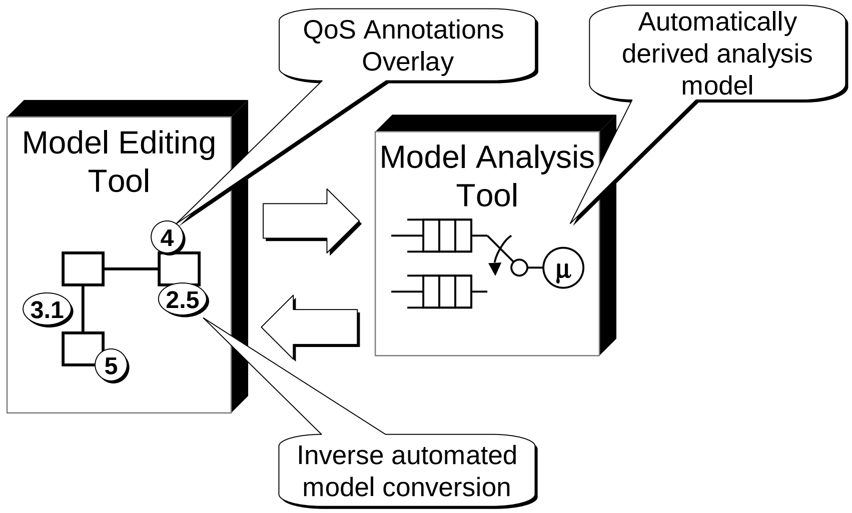
- Model transformations (e.g., code generation)
  - "Full" is where we want to be, "partial" is only a temporary solution
- Automated validation and verification
  - Formal methods (qualitative and quantitative)
  - Automated test generation and testing
- Computer-based model execution
  - Harel: non-executable models = "Engineless cars"
  - Particularly execution of abstract and incomplete models
    -- when most of the important decisions are made
  - Key to understanding and "feel"/intuition for problem
  - Also useful for buy-in to MDD ("Eureka! It works!")

Automatic Code Gen: State of the Art

- Efficiency
  - performance and memory utilization:
    within ±5-15% of equivalent manually coded system
- Scalability
  - compilation time (system and incremental change):
    within 5-20% of manual process
  - system size:
    - Complete systems in the order of 4MLOC have been constructed using full code generation
    - Teams of over 400 developers working on a common model

Automating Engineering Analysis

- Inter-working of specialized tools via shared standards

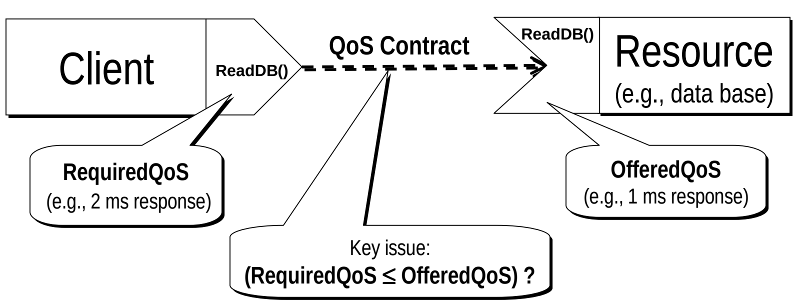


Concepts for Analyzing Models

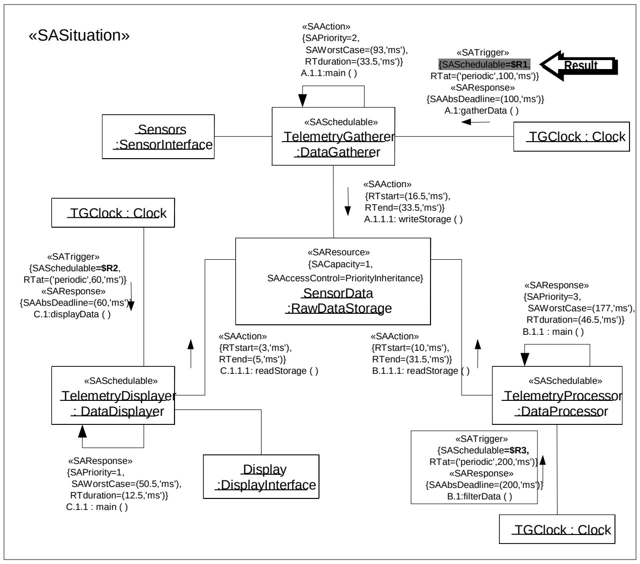
- *Resource*:

  an element whose ability or capacity is limited, directly or indirectly, by the finite capacities of the underlying physical elements

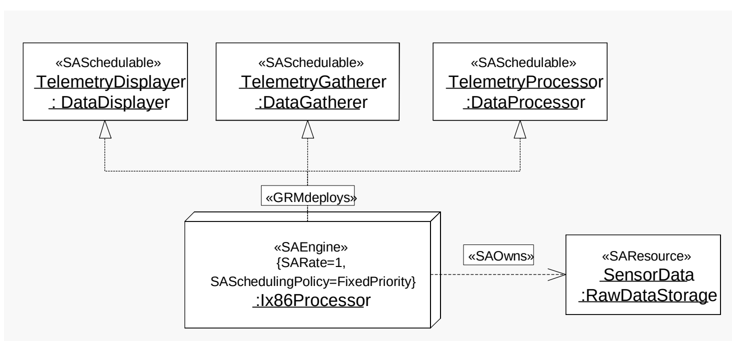
- The relationship between resources and resource users



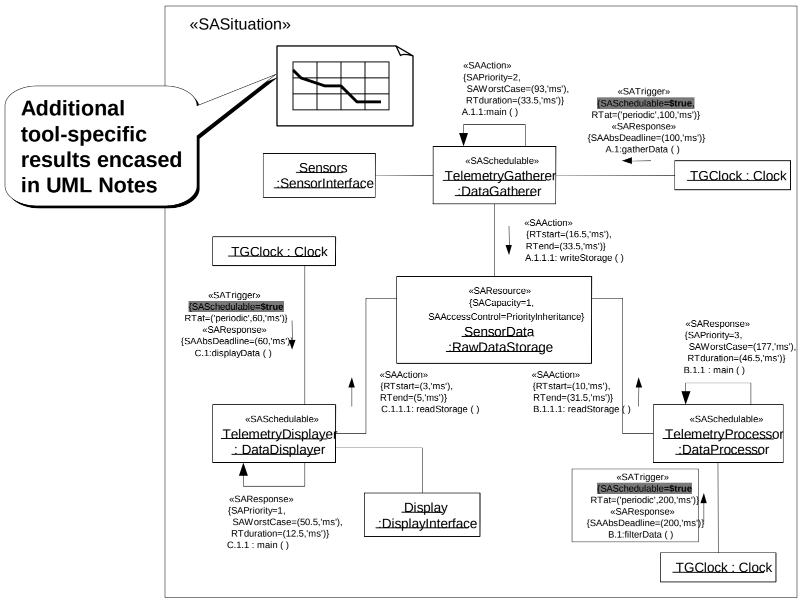
Example: Schedulability Annotations



Example: Deployment Specification



Example: Analysis Results

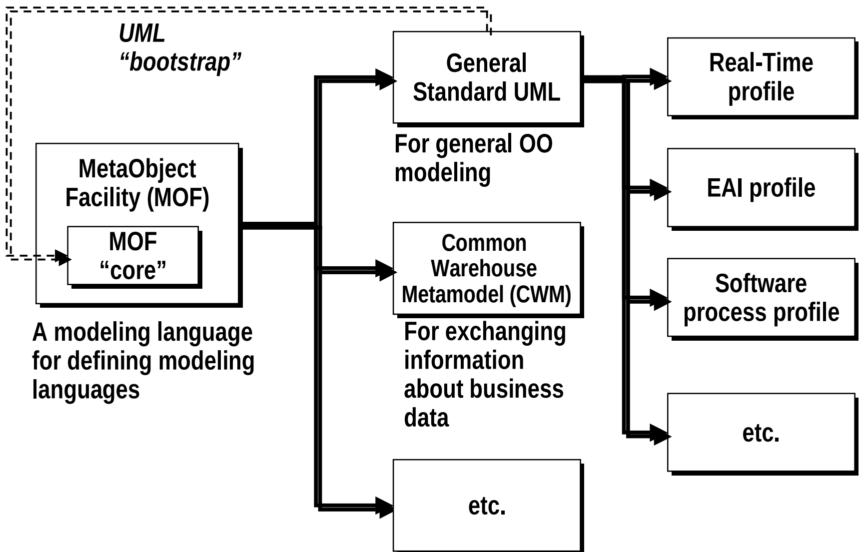


OMG's Model-Driven Architecture (MDA)

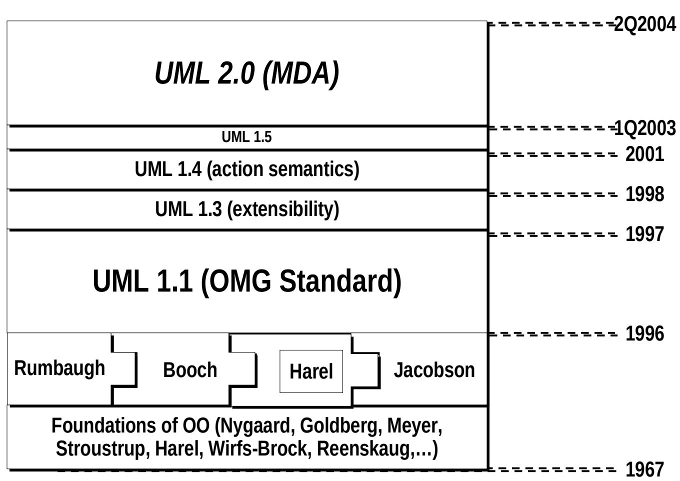
- An OMG initiative
  - A framework for a set of standards in support of MDD
- Inspired by:
  - The widespread public acceptance of UML and
  - The availability of mature MDD technologies
  - OMG moving beyond middleware (CORBA)
- Purpose:
  - Enable inter-working between complementary tools
  - Foster specialization of tools and methods
- Good overview paper:
  - http://www.omg.org/cgi-bin/doc?omg/2003-06-01

The Languages of MDA

- Set of modeling languages for specific purposes



UML: The Foundation of MDA

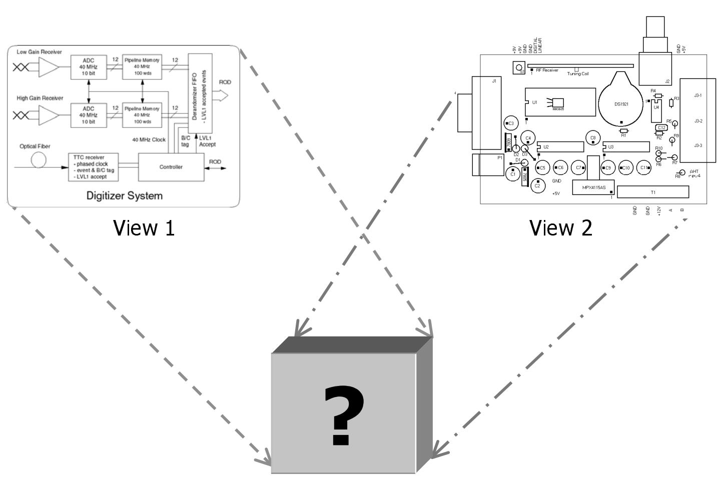


UML 1.x: What Went Right

- Timeliness (meeting a real need)
- Emphasis on semantics as opposed to notation
  - model-based approach (versus view-based)
  - detailed semantic specifications
- Higher-level abstractions beyond most current OO programming language technology
  - state machines and activity diagrams
  - support for specifying inter-object behavior (interactions)
  - use cases
- Customizability (extensibility)

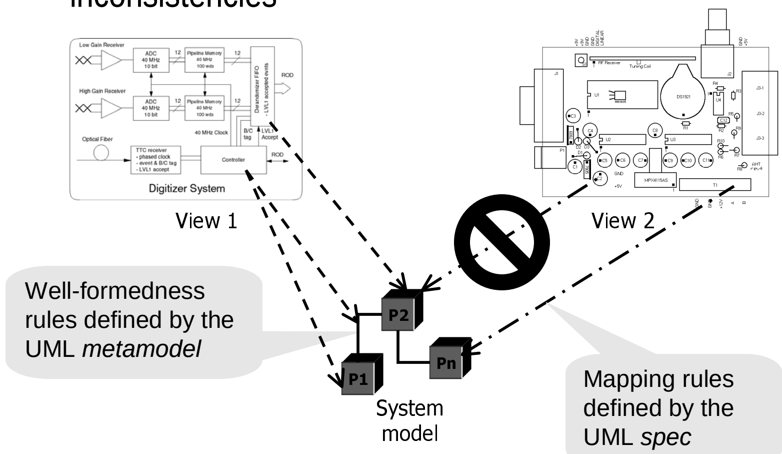
Traditional Approach to Views in Modeling

- Multiple, informally connected views
  - Combined in the final (integration) phase of design



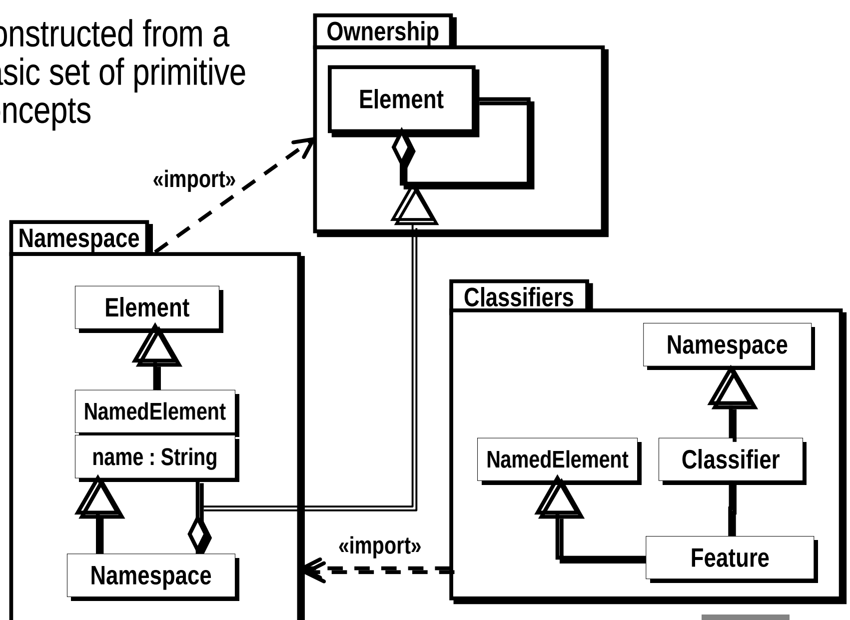
UML Approach: Single Model

- Views are projections of a complete model
  - Continuous integration of views with dynamic detection of inconsistencies



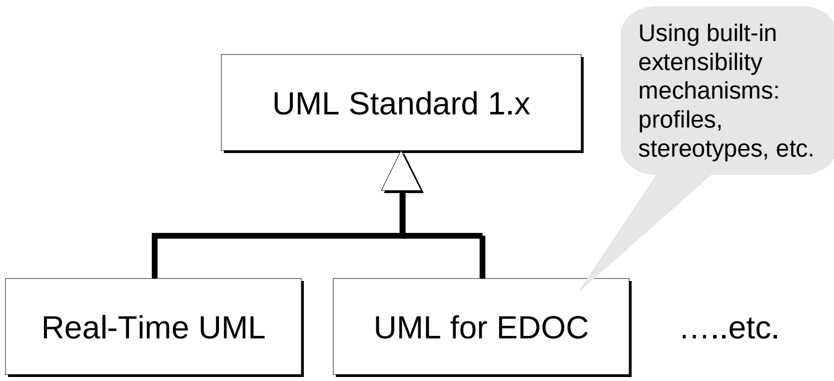
Metamodel Example: Defining Classifiers

- Constructed from a basic set of primitive concepts



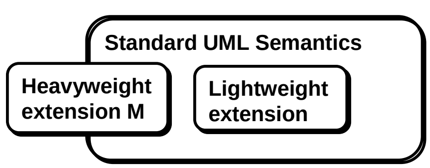
Specializing UML

- Avoiding the PL/I syndrome (“language bloat”)
  - UML standard as a basis for a “family of languages”



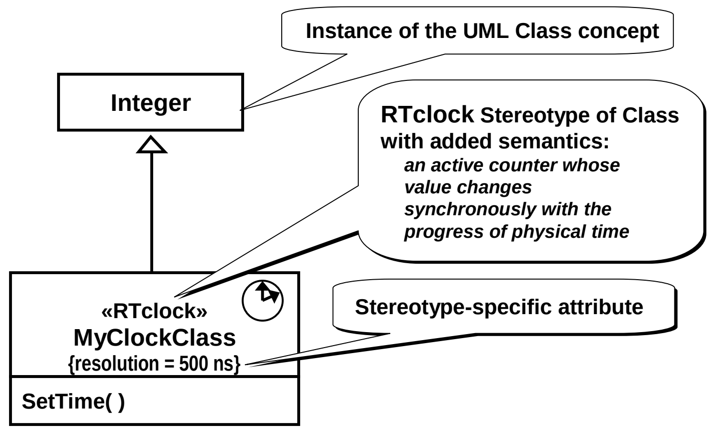
Specializing UML for a Domain

- *Lightweight extensions*
  - Extend semantics of existing UML concepts by specialization
  - Conform to standard UML (⇒ tool compatibility)
  - Profiles, stereotypes
- *Heavyweight (MOF) extensions*
  - Add new non-conformant concepts or
  - Incompatible change to existing UML semantics/concepts



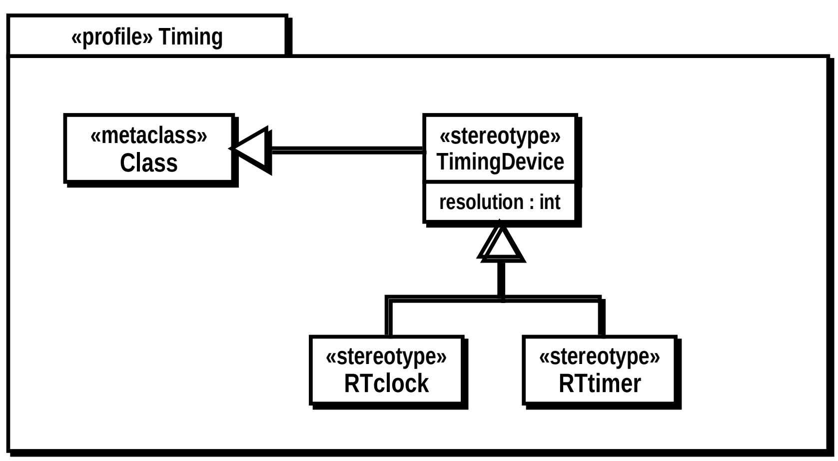
Stereotyping versus Inheritance

- For semantics not expressible through standard UML mechanisms
- Stereotypes can be standardized (application independent)



Creating UML Profiles

- E.g., specializing the standard Class concept



UML 1.x: What Went Wrong?

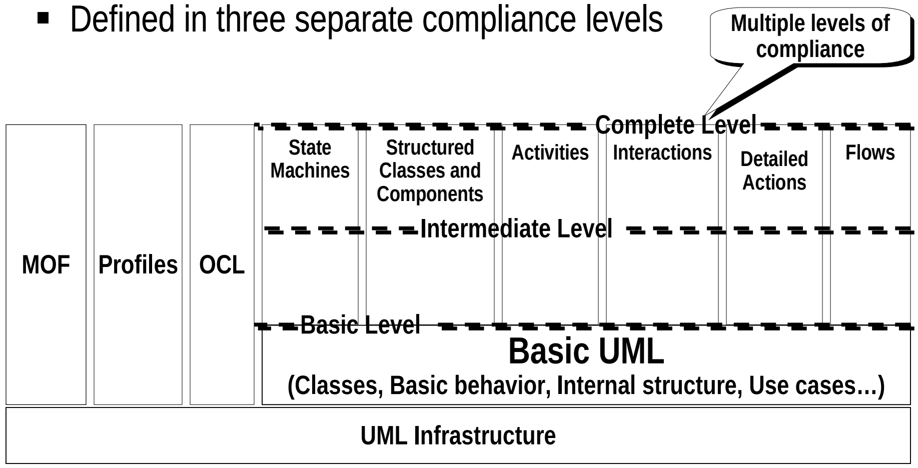
- Inadequate semantics definition
  - Vague or missing (e.g., inheritance, dynamic semantics)
  - Informal definition (not suitable for code generation or executable models)
- Does not fully exploit MDD potential of models
  - E.g., "C++ in pictures"
- Inadequate modeling capabilities
  - Business and similar processes modeling
  - Large-scale systems
  - Non-functional aspects (quality of service specifications)
- Too complex
  - Too many concepts
  - Overlapping concepts
- No diagram interchange capability
- Not fully aligned with MOF
  - Leads to model interchange problems (XMI)

UML 2.0 – an MDA Language

- Evolved out of UML 1.x
- Improved precision and semantics definition
- Small number of new modeling capabilities
  - Those required to support large and complex systems
- Backward compatibility with 1.x

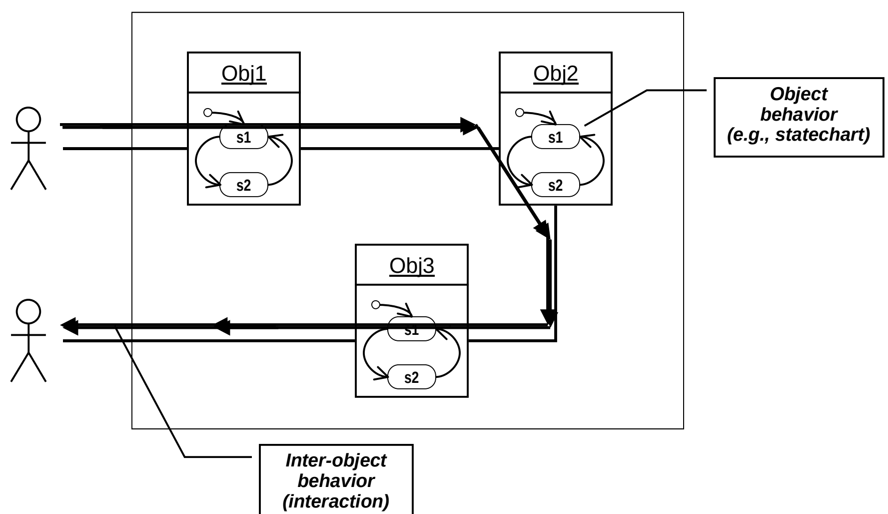
UML 2.0 Language Structure

- A core language + a set of optional “sub-languages”
  - Defined in three separate compliance levels

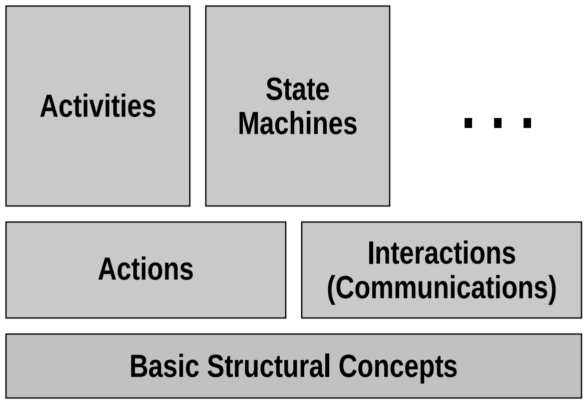


UML: Run-Time Semantics

- Causally-related event traces



Run-Time Semantics Architecture



Basic Structural Elements

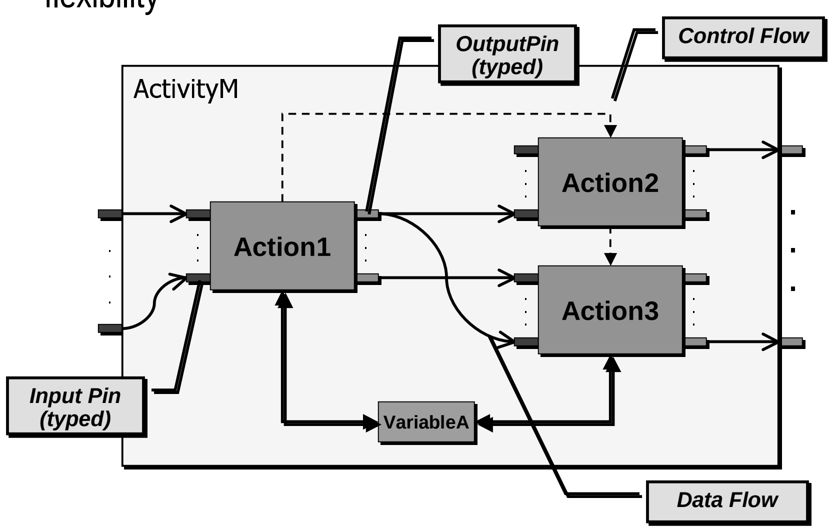
- Values
  - Universal, unique, constant
  - E.g. Numbers, characters, object identifiers (“instance value”)
- “Cells” (Slots/Variables)
  - Container for values or objects
  - Can be created and destroyed dynamically
  - Constrained by a type
  - Have identity (independent of contents)
- Objects (Instances)
  - Containers of slots (corresponding to structural features)
  - Just a special kind of cell
- Links
  - Tuples of object identifiers
  - May have identity (i.e., some links are objects)
  - Can be created and destroyed dynamically

Actions in UML

- Action = fundamental unit of behavior
  - For modeling fine-grained behavior (structure creation and manipulation actions, communications actions, control structures, etc.)
  - Level of traditional programming languages
- UML defines:
  - A set of action types
  - A semantics for those actions
    - i.e. what happens when the actions are executed
    - Pre- and post-condition specifications (using OCL)
  - No concrete syntax for individual kinds of actions (notation)
    - Flexibility: can be realized using different concrete languages
- In UML 2, the metamodel of actions was consolidated
  - Shared semantics between actions and activities

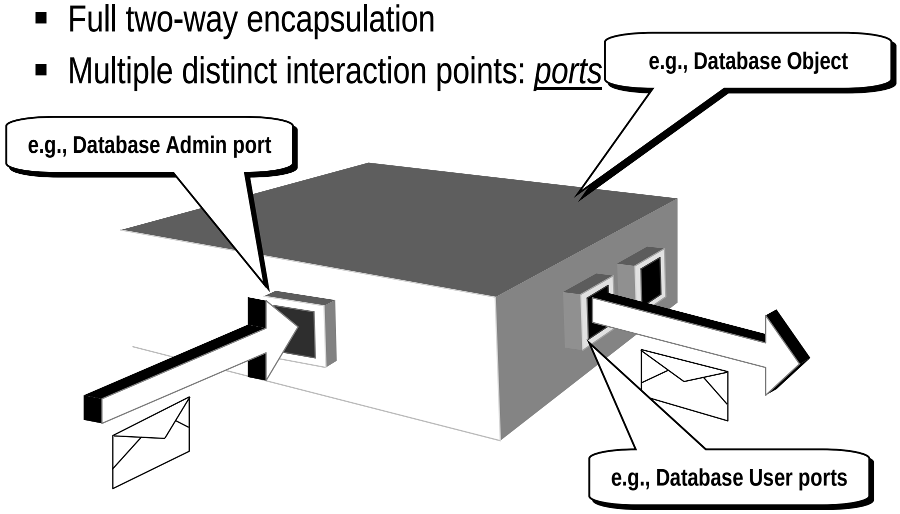
Shared Action/Activity Semantics

- Data/control flow foundations for maximal implementation flexibility



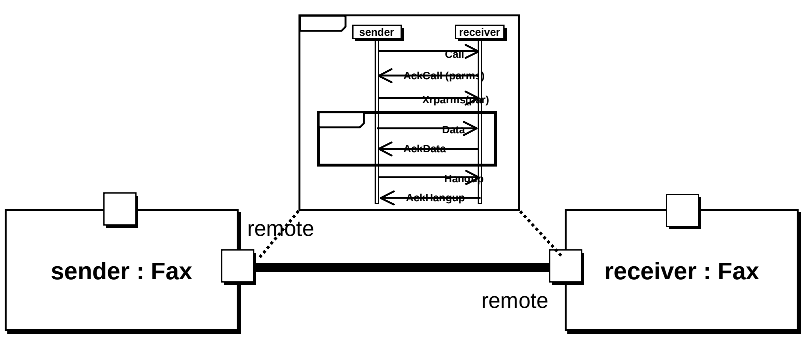
Structured Objects: External View

- Architectural-level objects with
  - Full two-way encapsulation
  - Multiple distinct interaction points: *ports*



Assembling Communicating Objects

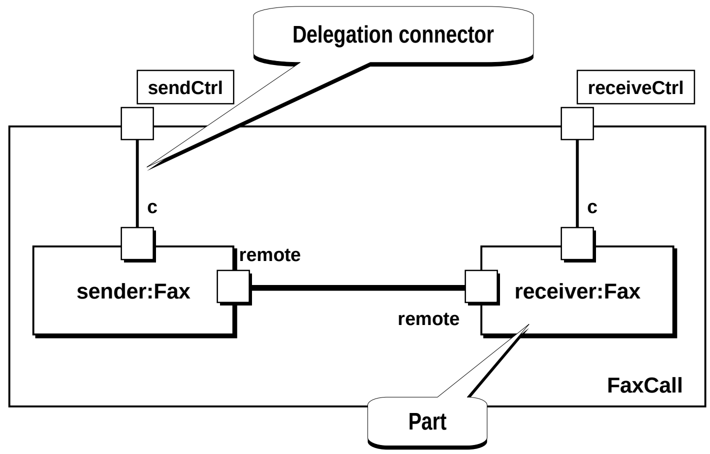
- Ports can be joined by *connectors* to create peer collaborations composed of structured classes



Connectors model communication channels
A connector is constrained by a *protocol*
Static typing rules apply (compatible protocols)

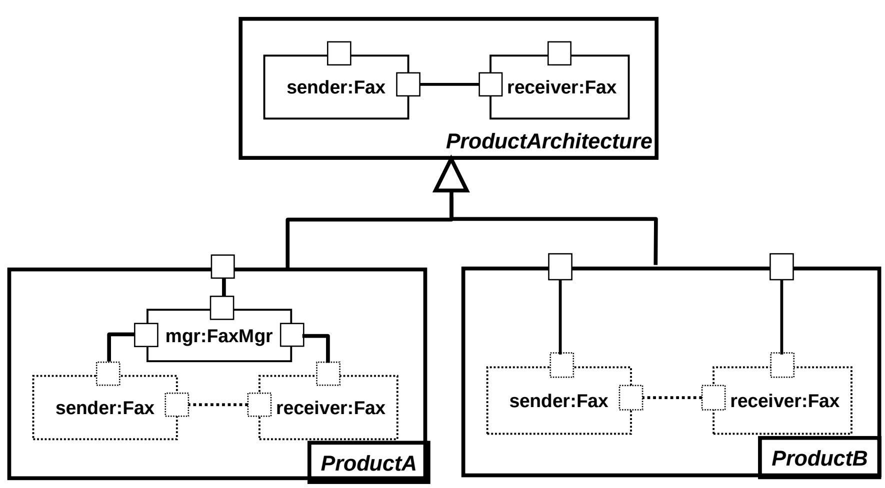
Structured Classes: Internal Structure

- Structured classes may have an internal structure consisting of structured class parts and connectors

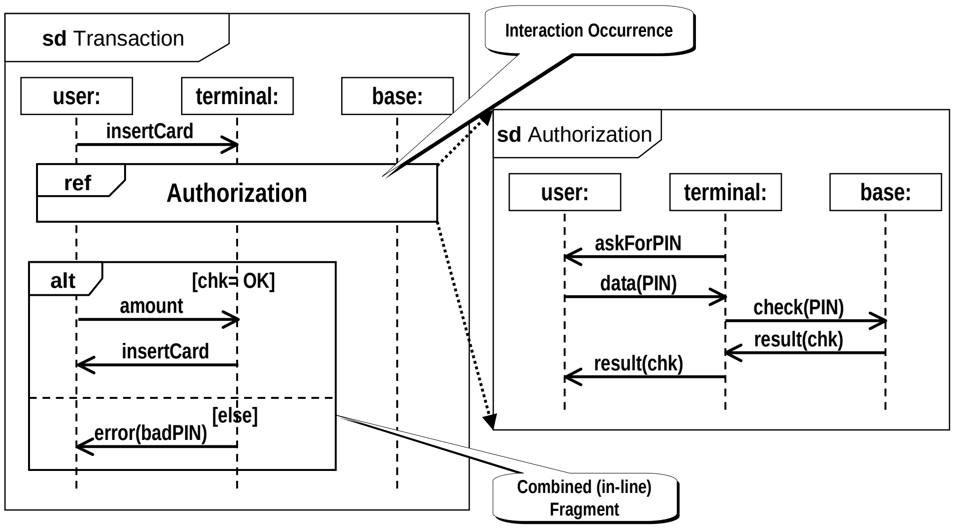


Product Family Architectures

- Using standard inheritance mechanisms!

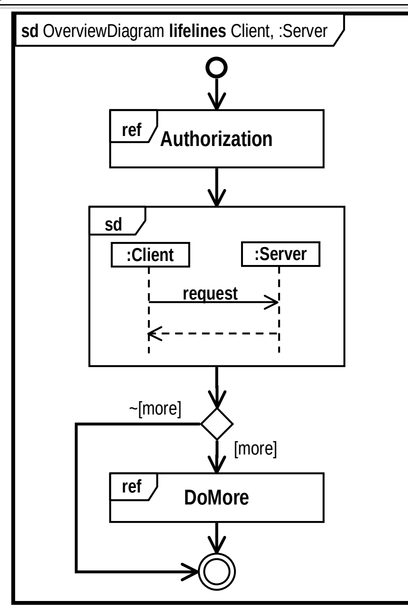


Interaction Diagrams



Interaction Overview Diagram

- Like flow charts
  - using activity graph notation for control constructs



What's being developed with UML

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, Operations And Maintenance, Optical Switching, Industrial Robot, Phone, Exchange (Pbx), 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

Summary: Model-Driven Development

- We cannot keep developing software the "traditional" way
  - Unreliable and slow for modern software needs
- MDD approach: increasing the levels of:
  - abstraction
  - automation
- Ready for prime time:
  - Field proven (major productivity and reliability gains)
  - Scalable MDD technologies (large industrial software projects)
  - Dedicated industry standards (OMG's MDA initiative, etc.)
- However, there is still a long way to go
  - Fertile area: many opportunities for cutting-edge research

QUESTIONS?

(bselic@ca.ibm.com)

School of Computing

Distinguished Seminar Series

Challenges for Bioinformatics Systems

Gregory Butler

Department of Computer Science
Concordia University
Thursday, February 26, 2004

Bioinformatics is a very diverse field, and as an applied field it is judged by how effectively it delivers tools to scientists working in genomics, proteomics, and other post-genomics areas of research. This demands advances in software systems, database systems, and intelligent systems in order to provide a foundation for rapid development of bioinformatics systems.

Our work at Concordia is addressing a range of issues across several key areas: generic C++ algorithm libraries for workstations and clusters; a database framework to provide technology for a variety of data models and allow intuitive data access for scientists; workflow and computational grids; ontologies, agents, and the semantic web; and usability of tools, web sites, and visualizations for bioinformatics.

Our active involvement in a large-scale fungal genomics project helps keep our focus firmly on the needs of scientists.

The seminar will provide a tour of our projects and highlight research issues. This is joint work with many colleagues at Concordia and within Quebec.

Biography

Gregory Butler is Professor of Computer Science at Concordia University, Montreal, Canada. His main research activities are methodologies for framework evolution, the development of a framework for databases and knowledge-bases, and applications to bioinformatics. Dr Butler is the author of over 50 technical papers. He has consulted on object-oriented design, object-oriented technology, database technology, and large-scale software architecture.

Dr Butler is a founder of the Centre for Structural and Functional Genomics in Montreal where he directs the development of the bioinformatics platform for a large-scale fungal genomics project.

Dr Butler obtained his PhD from the University of Sydney in 1980 for work on computational group theory. He worked in computer algebra for 20 years developing algorithms, constructing software systems, designing languages, and investigating the integration of databases and knowledge-bases with computer algebra systems. He as a major contributor to the Cayley/Magma system for computational group theory, modern algebra, and discrete mathematics.

Dr Butler was on the faculty of the Department of Computer Science at the University of Sydney from 1981 to 1990. He has held visiting positions in Delaware, Bayreuth, Karlsruhe, and Heidelberg.

Thursday, February 26, 2004
2:30-3:30 PM.
Walter Light WLH 210

The **Distinguished Seminar Series** is sponsored by the School of Computing and

CITO
Communications and Information Technology Ontario

Send comments about these Web pages to webmaster@cs.queensu.ca

Queen's University

School of Computing

Distinguished Seminar Series

Database Issues in Stream Data Management

M. Tamer Ozsu

University of Waterloo

Traditional databases store sets of relatively static records with no pre-defined notion of time, unless timestamp attributes are explicitly added. While this model adequately represents commercial catalogues or repositories of personal information, many current and emerging applications require support for on-line analysis of rapidly changing data streams. This talk will briefly review recent work in data stream management systems, and discusses our recent work in this area that focuses on sliding window multi-join processing in continuous queries over data streams. Several algorithms are reported for performing continuous, incremental joins, under the assumption that all the sliding windows fit in main memory. The algorithms include multi-way incremental nested loop joins (NLJs) and multi-way incremental hash joins. We also propose join ordering heuristics to minimize the processing cost per unit time.

Thursday May 6, 2004.
2:30-3:30 p.m.
Walter Light WLH 205

The **Distinguished Seminar Series** is sponsored by the School of Computing and

CITO
Communications and Information Technology Ontario

Better Connections.
Better Technology.
Better Results.

Queen's UNIVERSITY

School of Computing

Distinguished Seminar Series

Multipoint Communication in Optical Networks

[Ahmed E. Kamal]

Department of Electrical & Comp. Eng.
Iowa State University, Ames, Iowa

Several of the emerging applications in high-performance networks are of the multipoint traffic type. Since high-performance networks usually employ an optical network infrastructure, and since most of these applications require subwavelength bandwidth, several sessions are usually groomed on the same wavelength. Traffic grooming takes place in costly electronic equipment, such as SONET Add/Drop Multiplexers, or MPLS label switched routers. It is therefore important that such networks be designed in a way that is optimal in terms of cost, while also being able to effectively support this type of traffic. This talk will discuss the issues that arise in designing optical networks to support subwavelength multipoint traffic, and wil also present our recent research in this area. Optimal linear formulations for the network design problem will be presented. Because of the complexity of the problem, heuristic design techniques have been developed, and will also be presented.

Thursday June 3, 2004.
2:30-3:30 PM.
BIO 1102

The **Distinguished Seminar Series** is sponsored by the School of Computing and

CITO
Communications and Information Technology
Ontario