Modelling and Simulation

Fall Term 2003

General Information

Course title	**Modelling and Simulation**
Course number	**COMP 522A (CRN 1168) Fall Term 2003**
Prerequisites	CS 308-251 (data structures and algorithms), CS 308-302 (programming languages and paradigms), CS 308-350 (numerical computing). Some experience with object-oriented design and programming. If you do **not** have the pre-requisites for the course, the course will be **deleted** from your record by the Faculty of Science. This is a "project" course which means you will learn to build and use full modelling and simulation tools in the numerous assignments. You are strongly advised to not take **more than one other** "project" course.
Course venue	Trottier 1100 (check Minerva). Monday and Wednesday, 14:30 – 16:00
Enrollment Cap	Enrollment is **limited to 59** students.
Instructor	Prof. Hans Vangheluwe McConnell Engineering room 328 tel.: +1 (514) 398 44 46 e-mail: hv@cs.mcgill.ca
Office hours	Monday 16:00 – 18:00
TA	Thomas Feng
TA Office hours	Wednesday 16:00 – 18:00 (to be confirmed) McConnell Engineering room 202 (Modelling, Simulation and Design Lab)

Course Goals

Objectives

The course aims to teach the generic (*i.e.,* tool and application domain independent) concepts of modelling and simulation. By the end of this course, you should have a deep understanding of the concepts of modelling and simulation of dynamic systems using a variety of formalisms. You should be able to build modelling and simulation systems. This will give you ample background to understand and use existing modelling and simulation systems. The course presents general modelling and simulation principles by applying them to concrete problems. Through the assignments (building prototype modelling and simulation tools), some experience in structured, object-oriented design and implementation will also be acquired.

Prototype modelling tools and simulators are constructed based on the theory. This is preferred to using commercial simulation tools as it leads to much better insight.

Applications (software process modelling and simulation, reactive systems design such as complex graphical user interfaces, population dynamics analysis, traffic analysis, supermarket queueing, *etc.*) are used to illustrate the different modelling formalisms and serve as case studies for the tools built in the assignments.

Rationale and Content

In the course, a bird's eye view of the state-of-the-art in modelling and simulation is presented. Hereby, the close relationship between modelling and simulation on the one hand and the analysis and design of complex (software and hardware) systems is highlighted. A formal specification of modelling and simulation formalisms and processes reveals the need for a host of computer science techniques such as graph algorithms, compilers, computer algebra, software engineering, and graphical user interfaces. By means of these techniques, modelling and simulation tools are developed.

The modelling and simulation formalisms and tools are highly useful for the analysis, design, and implementation of complex, often embedded software systems, interacting with the physical world.

The complexity of current and future systems is not only due to a large number of components (tackled by hierarchical decomposition), but is also caused by the increasing diversity of components and problem aspects. A photocopier for example combines software, analog electronic, digital electronic, electrostatic, thermodynamic, hydraulic, ... aspects and components. To easily express the structure and behaviour of such systems, multiple formalisms must be used. Such models are a basis for documentation, analysis, formal proof, simulation what-if analysis, optimization and (embedded) application code generation.

The course presents a holistic view of the modelling and simulation enterprise. Rather than focusing on particular applications, or on specific tools, it starts from a general methodology which stresses the generic, application-independent aspects of modelling formalisms and their implementation. The main aim of the course is to provide the theoretical background, methods, techniques and tools for complex problem solving, with emphasis on the software aspects.

The formalisms covered range from Causal Block Diagrams, Differential Algebraic Equations, Forrester System Dynamics, to Finite State Automata, Statecharts, Petri Nets, DEVS, and the different Discrete Event World Views. More importantly, the relationships between these, as well as their relative merits and disadvantages, are investigated. For each formalism, the design and implementation of a *solver* or *simulation kernel* is presented.

From the practitioner's point of view, the course describes different modelling formalisms, existing languages and to a lesser extent, tools. From the computer scientist's point of view, the course describes the techniques and standards employed in the construction of modelling environments and simulators.

Each of the topics is introduced by means of an example, followed by a theoretical presentation, followed by a more in-depth example.

- Course Introduction. What is Modelling and Simulation. 1 lecture.
- Model Syntax and Semantics: Causal Block Diagrams. 3 lectures.
- System Specification Hierarchy, Model Classification. 3 lectures.
- State Automata and Petri Nets. 3 lectures.
- Higraphs and Statecharts. 3 lectures.
- Pseudo-random generators, input/output analysis. 1 lecture.
- Discrete Event World Views. 2 lecturs.
- Process Interaction. 3 lectures.
- DEVS. 3 lectures.
- Animation of simulation results. 1 lecture.

- Continuous-time models, solvers, sorting. 1 lecture.
- Population Dynamics, System Dynamics. 1 lecture.
- Analogy, Object-oriented Modelling of Physical Systems. 1 lecture.

Method of Instruction

Ex cathedra lectures.

Each group of new topics will be implemented in an assignment using the "executable pseudocode" scripting language Python.

Course Materials

There are no required texts.

Course material can be downloaded from the course homepage, and a selection of book chapters will need to be copied. For a couple of lectures, you will need to take notes in class.

Assignments and Evaluations

Assignments and Projects:

- 6 *assignments*. There are two types of assignments.
  One type of assignment consist of either the development of a small model (*e.g.,* of traffic behaviour at an intersection) in a particular formalism without simulation or of the use of an existing modelling and simulation system such as GPSS or PythonDEVS to obtain performance metrics through simulation.
  The other type of assignment includes *software development*. The rationale is that the Modelling and Simulation software tools to be developed are representative for "complex" software systems. During the development, most of the typical software engineering problems will be encountered. Obviously, the development of these tools will also provide insight into the (often abstract) Modelling and Simulation concepts.
- 1 *project* chosen from a list of proposed subjects. You can also propose your own subject. A project should be chosen and started before the first week of October. Intermediate deliverables must be presented. At the end of term, a report and all developed code must be available on the WWW, and a formal presentation must be given in class.

Structure:

- The assignments must be done in groups of 2 members. Both team members must work on the assignment together ("pair work") and hence will both be able to explain the assignment solution.
- The results of some assignments will be presented and discussed in class.
- All assignments must be submitted entirely in the form of web-pages (including design, code, and simulation results) via WebCT.
- The project may be worked on in teams of upto 3 members, though this is not necessary. *Individual work must be indicated* !

Grading

Grades will be distributed over assignments, project and exam:

- 70% on 6 assignments.

- 30% on the project.
- There is *no final exam* as together, assignments and project will cover the entire course.

Students with marks of D, F or J will have the option of doing Additional work to improve assignment/project grades. Note how this extra work is treated as a *supplemental exam* and requires obtaining faculty permission !

Assignments and projects will be judged on:

- correctness,
- structure and completeness,
- amount of work,
- originality,
- presentation.

Original Work

You are encouraged to help each other formulate the ideas behind assignment problems, but each student is required to submit his or her own *original* work. Handing in work that is not your own, original work as if it is your own is plagiarism.

McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the code of student conduct and disciplinary procedures (see www.mcgill.ca/integrity for more information).

References

Background information for the course can be found in:

- A plethora of modelling formalisms: Paul A. Fishwick. Simulation Model Design and Execution, building digital worlds: Prentice Hall, 1995.
- The foundations of modelling and simulation: Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, second edition, 2000. Chapters 1 – 9, 17, 18.
- Random variates, random number generation: Averill M. Law and David W. Kelton. Simulation Modeling and Analysis. McGraw-Hill, 1991. Chapters 8, 9.
- Discrete event world views: Osman Balci. The implementation of four conceptual frameworks for simulation modeling in high-level languages. In M. Abrams, P. Haigh, and J. Comfort, editors, Proceedings of the 1988 Winter Simulation Conference, pages 287–295. Society for Computer Simulation International (SCS), 1988.
- The process interaction language GPSS: Geoffrey Gordon. System Simulation. Prentice Hall of India, second edition, 1996. Chapters 8 – 10.
- Continuous system modelling theory, causality, Forrester System Dynamics: Francois E. Cellier. Continuous System Modeling. Springer-Verlag, New York, 1991. Chapters 1, 2, 5, 7, 10, 11, 15.
- Numerical simulation, System Dynamics: Hartmut Bossel. Modeling and Simulation. A.K. Peters, Ltd., 289 Linden Street, Wellesley, MA 02181, 1994. Chapters 1 – 3.
- Petri Nets and Timed Models: Christos G. Cassandras. Discrete Event Systems. Irwin, 1993. Chapters 4, 5.
  Tadao Murata. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 77(4):541–580, April 1989.
- Statecharts and applications in object-oriented software design: David Harel. On visual formalisms. Communications of the ACM, 31(5):514–530, May 1988.

David Harel and Eran Gery. Executable object modeling with statecharts. IEEE Computer, pages 31–42, 1997.

- Object-oriented non-causal modelling: Hilding Elmqvist et. al. Modelica – a unified object-oriented language for physical systems modeling: Tutorial and rationale. The Modelica Design Group, December 1999. http://www.modelica.org/.
  Robert Endre Tarjan. Data Structures and Network Algorithms. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial & Applied Mathematics, 1983. Chapter 8. (used for causality assignment)

All the above are available at the Schulich library.