Systems Engineering:
Multi-formalism Modelling and Simulation

Fall Term 2000

General Information

Course number	CS 308-767A
Course title	Multi-formalism Modelling and Simulation
Prerequisites	Set theory, numerical analysis, object-oriented programming. Compilers background is appreciated but not necessary. Do **not** take 767A in conjunction with **more than one** other "project" course.
Course venue	McConnel Engineering room 320 Monday and Wednesday, 13:00 – 14:30 (starting Wed 6 September)
Instructor	Prof. Hans Vangheluwe McConnell Engineering room 328 tel.: +1 (514) 398 44 46 e-mail: hv@cs.mcgill.ca
Office hours	Monday 14:30 – 18:30 (formulate questions via e-mail first)

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 using a variety (and combination) of formalisms. You should be able to build, from scratch, modelling and simulation systems (during the course only two are used, others are built). This will give you ample background to understand and use existing modelling and simulation tools. The course wants to you for the application of general modelling and simulation principles to concrete problems. Through the assignments (building prototype modelling and simulation tools), some experience in structured, object-oriented design and implementation should be acquired.

The course's main focus is *not*:

- how to use simulation tools. By implementing your own prototypes based on the theory, a much deeper insight will be gained;
- how to apply modelling and simulation in specific disciplines. Applications (software process modelling and simulation, reactive systems design, population dynamics analysis, mechatronics, supermarket queueing, *etc.*) are used to illustrate the different modelling formalisms and serve as use cases 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 (and possibly a hint of the future) 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 supporting their use are themselves 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, multi-formalism models can 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, and Bond Graphs, to Finite State Automata, State Charts, Petri Nets, DEVS, and the different Discrete Event World Views. More importantly, the relationships between these, how to meaningfully couple them as well as their relative merits and disadvantages, are investigated. For each formalism, the design and –possibly distributed– 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.

Topics and Schedule

An overview of topics covered is presented with the schedule of classes.
As the course progresses, this overview will be filled in with links to presentation material, online articles, assignments, *etc.*

Method of Instruction

Ex cathedra lectures, with (hopefully) frequent interaction from the audience.

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. Before/during each lecture, both notes and articles will be made available, in electronic form whenever possible.

The most relevant books are:

- The theory of modelling and simulation is presented in Zeigler [10]. This is the second, and completely

re-worked editon of his seminal work, first published in 1976 [9]. The new edition is the foundation for any development in modelling and simulation. It has a strong focus on DEVS (Discrete Event System Specification). We will cover at most 25% of the book (tempus fugit, alas).
- Continuous modelling, including non-causal modelling and Bond Graphs is covered in Cellier [2]. Causal modelling and simulation, in particular System Dynamics style modelling is presented through a plethora of examples in Bossel [1]. Code for the numerical analysis used in continuous simulators is found in Numerical Recipes [8].
- A standard reference for (mainly) discrete event simulation is Law and Kelton [5]. We refer to it for its treatment of pseudo-random number generators and statistical analysis of results. The (out of print, except in Indian edition) System Simulation book by Gordon [3] presents the still popular General Purpose Simulation System (GPSS) representative for the Process Interaction world view.
- All examples as well as programming assignments will use the scripting language Python. A host of information on Python is available online at www.python.org. Learning Python [6] is a thorough introduction to Python. Programming Python [7] goes into far more detail and describes how to embed and extend Python. To develop GUIs in Python (with TkInter), Grayson [4] provides an in-depth framework.

With the exception of Cellier's book (to be ordered), the above are available at the Physical Sciences and Engineering Library (PSEL).

A list of (suggested) references to articles and websites is found on each lecture's page.

Assignments and Evaluations

Assignments and Projects

- 4 small exercises (typically, the use of an existing modelling and simulation system, *e.g.*, GPSS exercises).
- 4 large *software development* assignments. The rationale is that the Modelling and Simulation software tools to be developed in the projects 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, similar to an assignment, but larger. It is typically an elaboration of a previous assignment. A project should be chosen and started as soon as possible.

Some assignments and the project may be worked on in teams (of 2 or 3 members), though this is not necessary. Individual work must be indicated.

Grading

Assignments and projects will be judged on:

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

Grades will be distributed over assignments and project/exam:

- 60% on assignments. Weighting of individual assignments is based on difficulty and amount of work. Large assignments carry at least 5 times as much weight as small exercises (*e.g.*, GPSS mini-assignments).

- 40% on the project. If a final theory exam is preferred (instead of a project – the default option for this course), only those parts of the course for which no assignments were made are subject matter.

It is expected that all students understand University policies on academic honesty.

References

[1] BOSSEL, H. *Modeling and Simulation*. A.K. Peters, Ltd., 289 Linden Street, Wellesley, MA 02181, 1994.

[2] CELLIER, F. E. *Continuous System Modeling*. Springer-Verlag, New York, 1991.

[3] GORDON, G. *System Simulation*, second ed. Prentice Hall of India, 1996.

[4] GRAYSON, J. E. *Python and Tkinter Programming*. Manning Publications, 2000.

[5] LAW, A. M., AND KELTON, D. W. *Simulation Modeling and Analysis*. McGraw-Hill, 1991.

[6] LUTZ, M. *Programming Python*. O'Reilly & Associates, 1996.

[7] LUTZ, M., AND ASCHER, D. *Learning Python*. O'Reilly & Associates, 1999.

[8] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C, The Art of Scientific Computing*, second ed. Cambridge University Press, 1992.

[9] ZEIGLER, B. P. *Theory of Modelling and Simulation*. Robert E. Krieger, Malabar, Florida, 1984.

[10] ZEIGLER, B. P., PRAEHOFER, H., AND KIM, T. G. *Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, second ed. Academic Press, 2000.