CS 308-767A
Systems Engineering:
Multi-formalism Modelling and Simulation

Hans Vangheluwe

Fall Term 2000

Lectures

Lecture 1 – Wednesday, September 6

What is *Modelling and Simulation* ?

Modelling and Simulation for Software Development for Modelling and Simulation for . . .

Lecture 2 – Monday, September 11

A simple mass-spring example demonstrating the full modelling and simulation process.
This includes the design of a simple time-slicing simulator.

The Modelling and Simulation process.

Lecture 3 – Wednesday, September 13

The use of scripting languages (Python) for:

- Rapid Application Development, prototyping (object-oriented executable pseudo-code).
- Multi-platform development (interpreted byte-code).
- Automated software testing.
- Glueing heterogeneous components.
- Scripting, embedding, extending traditional code.

Assignment: Block diagram modelling and Time-slicing simulation

Lecture 4 – Monday, September 18

Classification of model types.

Multi-abstraction modelling: the behaviour morphism.

Hierarchy of System Specification.

The concept of *state*, state *trajectories*.

Lecture 5 – Wednesday, September 20

General *systems theory* for causal, deterministic models, with examples.

Classification re-visited. *Multi-formalism modelling*.

Lecture 6 – Monday, October 2

(Finite) State Automata (Moore, Mealy).

Reading material:

- Christos G. Cassandras. *Discrete Event Systems*. Irwin, 1993.

Lecture 7 – Wednesday, October 4

Adding *synchronisation* and *concurrency* to FSA: *Petri Nets*.
Symbolic analysis of Petri Nets.
The PNS simulator.

Reading material:

- Christos G. Cassandras. *Discrete Event Systems*. Irwin, 1993.
- Tadao Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580. April 1989.

Assignment: A Finite State Automata AND Petri Net Modelling, Simulation, and Visualization tool

Lecture 8 – Wednesday, October 11

Adding *hierarchy* and *orthogonality* to Finite State Automata: *Higraphs, State Charts*.

State Charts for the analysis and design of systems: Executable Object Modelling.

Reading material:

- David Harel. On Visual Formalisms. *Communications of the ACM*, 31(5):514–530. May 1988.
- The *Unified Modelling Language* (UML) and Executable Object Modelling.
  David Harel and Eran Gery. Executable Object Modeling with Statecharts. *IEEE Computer*, July 1997. pp. 31–42.
- Process Programming: Entity Process Modelling (EPM).
  Watts S. Humphrey and Marc I. Kellner. *Software Process Modeling: Principles of Entity Process Models*.
  Technical Report CMU/SEI-89-TR-002. 23 pp. February 1989.

Implicit assignment: in all assignments, GUI behaviour must be described by means of State Chart models.

Lecture 9 – Monday, October 16

Discrete Event simulation concepts.

World Views and their simulation kernels:

- Event Scheduling;
- Activity Scanning;
- Three Phase Approach;
- Process Interaction (intro).

Deterministic simulation of *stochastic* processes.
Introduction to (pseudo) *Random Number Generators*.
Generating arbitrary distributions.
Testing of pseudo random number generators.

C code for a prime modulus multiplicative linear congruential generator.

Gathering *statistics* (performance metrics).

Reading material:

- Osman Balci. The implementation of four conceptual frameworks for simulation modeling in high-level languages.
  Proceedings of the 1988 Winter Simulation Conference. pp. 287–295. Society for Computer Simulation/Association for Computing Machinery. 1988.

Lecture 10 – Wednesday, October 18

Process Interaction Modelling practice: GPSS, basic concepts and blocks.

Material (also for next class):

- Geoffry Gordon. *System Simulation*. Prentice Hall. Second Edition. 1996. Chapters 9–10.
- Online resources: GPSS/H specific information, Gordon's examples in GPSS/H
- Peter Lorenz's excellent Simulation and Animation course (University of Magdeburg, Germany)

Mini assignments: *all* Gordon's assignments (Chapter 9).

Lecture 11 – Monday, October 23

Process Interaction Modelling practice: GPSS, advanced concepts.

Mini assignments: *all* Gordon's assignments (Chapter 10).

Lecture 12 – Wednesday, October 25

Process Interaction Modelling practice: queueing disciplines: user chains (the LINK and UNLINK blocks in detail).

Lecture 13 – Monday, October 30

Process Interaction Modelling practice: GPSS, the simulator (semantics in terms of operations on CEC and FEC data structures).

Process Interaction Modelling in πDemos (*formally* described operational semantics).

Reading material:

- Peter Lorenz's notes (based on Schriber) on GPSS/H's Internal Control
- G. Birtwistle and C. Tofts. Operational Semantics of Process-oriented Simulation Languages. Part 1: πDemos.
  Transactions of the Society for Computer Simulation, 10(4). July 1994. pp. 299–333.

Lecture 14 – Wednesday, November 1

DEVS, a foundation for Discrete Event Modelling and Simulation: atomic DEVS

Mini assignment: DEVS model of a single server, single queue system with a priority queue and "president" interrupt.

Lecture 15 – Monday, November 6

DEVS, a foundation for Discrete Event Modelling and Simulation:

- coupled DEVS, closure
- simulators and coordinators

Lecture 16 – Wednesday, November 8

Jean-Sebastien Bolduc's presentation of his DEVS simulator in Python (simulator + examples).

Lecture 17 – Monday, November 13

Pseudorandom number generators (see presentation October 16).

Models vs. visualization vs. animation: models as the basis for the semantics of visual representation.

Real-time simulation and animation.

Causal *continuous-time* models: use, mathematical representation.

Continuous *solvers*: structure, error-control, stiff systems, algebraic loops, DASSL.

Lecture 18 – Wednesday, November 15

Algebraic *loop detection* and *sorting*.

Continuous System Simulation Languages (CSSLs).

A *neutral model-solver architecture* (API), experimentation. Simulation Experiments:

- Optimisation, shooting, parameter estimation.
- Experiments as scripts.
- Experimental Frames.

Lecture 19 – Monday, November 20

Hybrid modelling: combining continuous and discrete abstractions.

Introducing *discontinuities*: abstracting reality, approximation by fast dynamics, piecewise continuous mathematical representation, zero-crossing location in solvers.

Lecture 20 – Wednesday, November 22

Putting it all together: an Event Scheduling Hybrid Differential Algebraic simulation kernel.

Lecture 21 – Monday, November 27

Population Dynamics:

- deductive, physical modelling vs. inductive modelling.
- Growth and decay.

- Predator-prey models.
- Competition and cooperation.

Lecture 22 – Wednesday, November 29

The Forrester System Dynamics *process*, introducing *inductive modelling* (structure identification).

WEST++ for System Dynamics.

Lecture 23 – Monday, December 4

Object-Oriented Modelling of Physical Systems.

- The need for non-causal modelling. Causality Assignment: maximum cardinality matching.
- Computer Algebra basics: canonical representation, symbolic algorithms $\neq$ numerical algorithms.
- Scoping, OO modelling vs. OO simulation. Object-oriented model specification:
  - Modelling language requirements.
  - The Modelica Unified Object-oriented Language for Physical Systems Modelling

  Semantic checks in the modelling language (*dynamics*) versus at the *graphical* modelling level.

Lecture 24 – Wednesday, December 6

Object-Oriented Modelling of Physical Systems: *Bond Graphs*. "Bond Graphs" is a non-causal, domain-independent (unifying electrical, mechanical, hydraulical, . . . systems) formalism.

- Physical Analogy. The transformation between analog and similar formalisms. Analog computers.
- The Idealised Physical Modelling process.

Putting it all together: multi-formalism modelling and simulation.

- Coupled models and hierarchy: closure under coupling, to flatten or to coordinate ?
- The Formalism Transformation Graph (FTG). The FTG describes possible mappings of one formalism onto another.
- Multi-formalism model flattening, transformation, and optimization: the algorithms.

Present and discuss assignment/project results.

Project Suggestions

All are software projects, with the exception of mapping GPSS onto DEVS, which is pen-and-paper (well, LaTeX) only.

1. A hierarchical, multi-formalism, graphical model editor.
2. A higraph graphical editor (for State Charts).
3. Generating reactive code from State Charts.
4. A GPSS (subset) modelling and simulation environment.
5. Mapping Process Interaction Formalism constructs (*i.e.*, a subset of GPSS) onto DEVS.
6. A parallel DEVS simulator (based on Jean-Sébastien Bolduc's DEVS simulator in Python).
7. A hybrid simulation kernel (based on Pai Hsueh-Ieng's prototype).
8. A non-causal model transformer (using Dinic's network flow algorithm).
9. A Bond Graph modelling tool.