CoSMoS: A Reusable Approach to Complex Systems Simulation

Adam T. Sampson
School of Computing, University of Kent

The CoSMoS project

- Developing an engineering process for the construction and use of complex systems simulations across all fields of scientific experimentation
- EPSRC-funded 4-year interdisciplinary project spanning multiple institutions

CoSMoS

Complex Systems Modelling and Simulation infrastructure

Some terminology

- Complex systems consist of many interacting components, and often exhibit emergent behaviours
- Models are abstractions used to describe and reason about a system
- Simulations are executable models that can be used to perform experiments

Who's involved?

- York
  - Susan Stepney
  - Jon Timmis
  - Andy Tyrrell
  - Fiona Polack
  - Paul Andrews
- Abertay
  - Jim Bown
- UWE/BRL
  - Alan Winfield
- Kent
  - Peter Welch
  - Fred Barnes
  - Adam Sampson
- Microsoft Research
- Celoxica

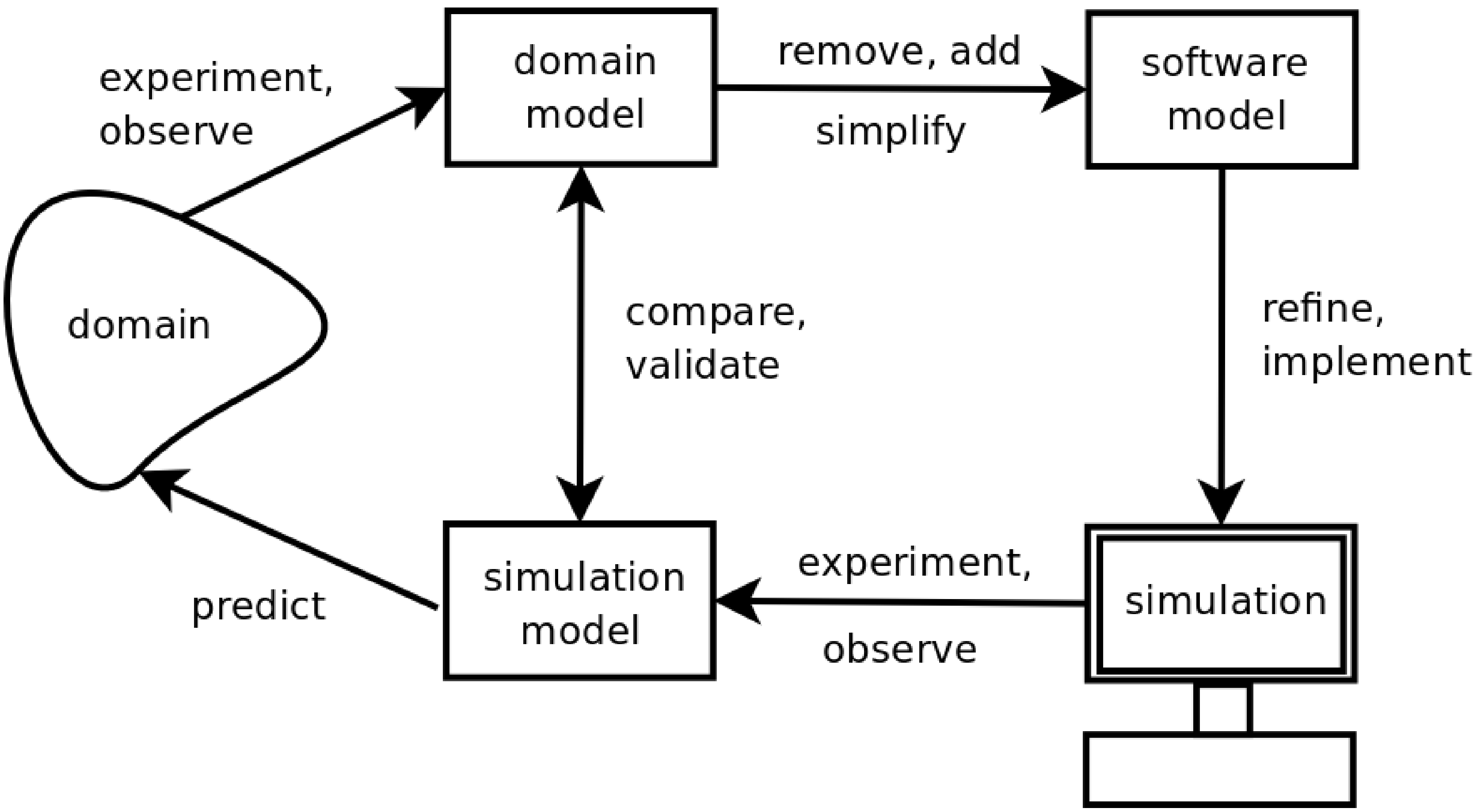
Why CoSMoS?

- At the moment, most researchers build simulations from scratch in an ad-hoc fashion
- This causes various problems:
  - Duplicated effort
  - Inefficient implementation
  - Hard to reason about simulation validity
  - Peer review of your paper won't find errors in your simulation...

The CoSMoS process

- The CoSMoS process is an agile approach based on design patterns
  - Select the patterns that suit your particular problem
- Patterns for all stages of development: design, implementation, application, analysis, validation
- Document assumptions at all stages
- Defined roles
  - How to interact effectively with your domain expert
- A library of reusable software components

Different models



Collecting patterns

- Our approach is driven by case studies
- We're implementing and experimenting with many different case studies, and extracting the useful common elements as design patterns
- Several sources of case studies:
  - standard textbook examples (e.g. bird flocking)
  - interesting papers (e.g. Amos' annular sorting)
  - real applications from CoSMoS partners (e.g. plants, lymphocyte rolling, granuloma formation, social exclusion...)

Building simulations

- The simulations we've implemented so far are all *agent-based*, with agents implemented as lightweight threads (*processes*)
- We aim for *scalability* over straightline performance
  - Many emergent systems require large populations to produce interesting behaviours
  - Some overhead is acceptable if it lets you run bigger simulations by using multiple CPUs/machines

Case study: bird flocking

- Our first case study: Reynolds' boids
- Birds form flocks through simple rules:
  - Match velocity with birds around you
  - Move towards centroid of birds around you
  - Avoid collisions
- Developed reusable approach to modelling continuous space; reused in later case studies

Boids demo

Distributing a simulation

- Our simulation ran on a single machine to start with – but we want to go bigger!
- Divide the simulation across multiple machines...
- ... and have agents migrate between machines as they move around in the world
- Developed refactoring patterns for doing this efficiently without changing the behaviour of the simulation; reused in later case studies

The Tromsø Display Wall

Birds on the Wall

- The Display Wall at the University of Tromsø
- 28 (7x4) fast Linux machines, each with a projector attached
- Aimed at distributed processing and scientific visualisation – very high resolution display
- A variety of mechanisms for interaction – gesture tracking, 3D sound location...
- We ported some of our simulations to run on the Wall, adding support for interaction

Birds on the Wall video

Case study: lymphocyte migration

- Our first real application
- HEVs are sections of blood vessel inside lymph nodes, where lymphocytes pass from the blood system into the lymphatic system
- Lymphocytes collide with the vessel wall, “roll” along it, then migrate through into the lymph node
- York researchers wanted to experiment with how the dilation of the HEV (upon infection) affects the rate at which migration occurs...

Lymphocyte demo

Case study: granuloma formation

- Another application: studying the formation of granulomas – clusters of cells that fight infection
- We model the network of blood vessels (sinusoids) in one lobule of the liver
- Chemokine signals diffuse away from infected regions
- Kupffer cells follow the chemokine gradient to locate and destroy pathogens
- Work in progress...

Granuloma demo

Irregular concurrency

- Our simulations are not trivially-parallelisable; they have a high degree of *irregular concurrency*
- Agents behave unpredictably with regard to when and with whom they communicate
- We need to be able to express this kind of concurrency in our programs...
- ... and have a runtime system that can execute these sorts of programs efficiently
  - Expose the concurrency in your program, and you get parallelism for free

Process-oriented programming (POP)

- A practical approach to concurrent design, based on the ideas of CSP and the pi-calculus
  - This allows static checking of the behaviour of programs – e.g. freedom from deadlock
- Isolated *processes* that communicate by passing messages along *channels*
- Ideally suited for things with irregular concurrency – e.g. agent-based simulations, network programming
- Many implementations (Google's Go, occam-pi, JCSP, PyCSP, CSO, CHP...)

CCSP concurrent runtime system

- Lightweight processes
  - Only limited by memory – 8 words per process
  - Hundreds of thousands of agents per machine
- World-class performance
  - Context switch in the <100ns range
- Automatic load-balancing for multicore CPUs
  - Batch processes based on communications
  - Minimise cache contention
- Supports multiple languages

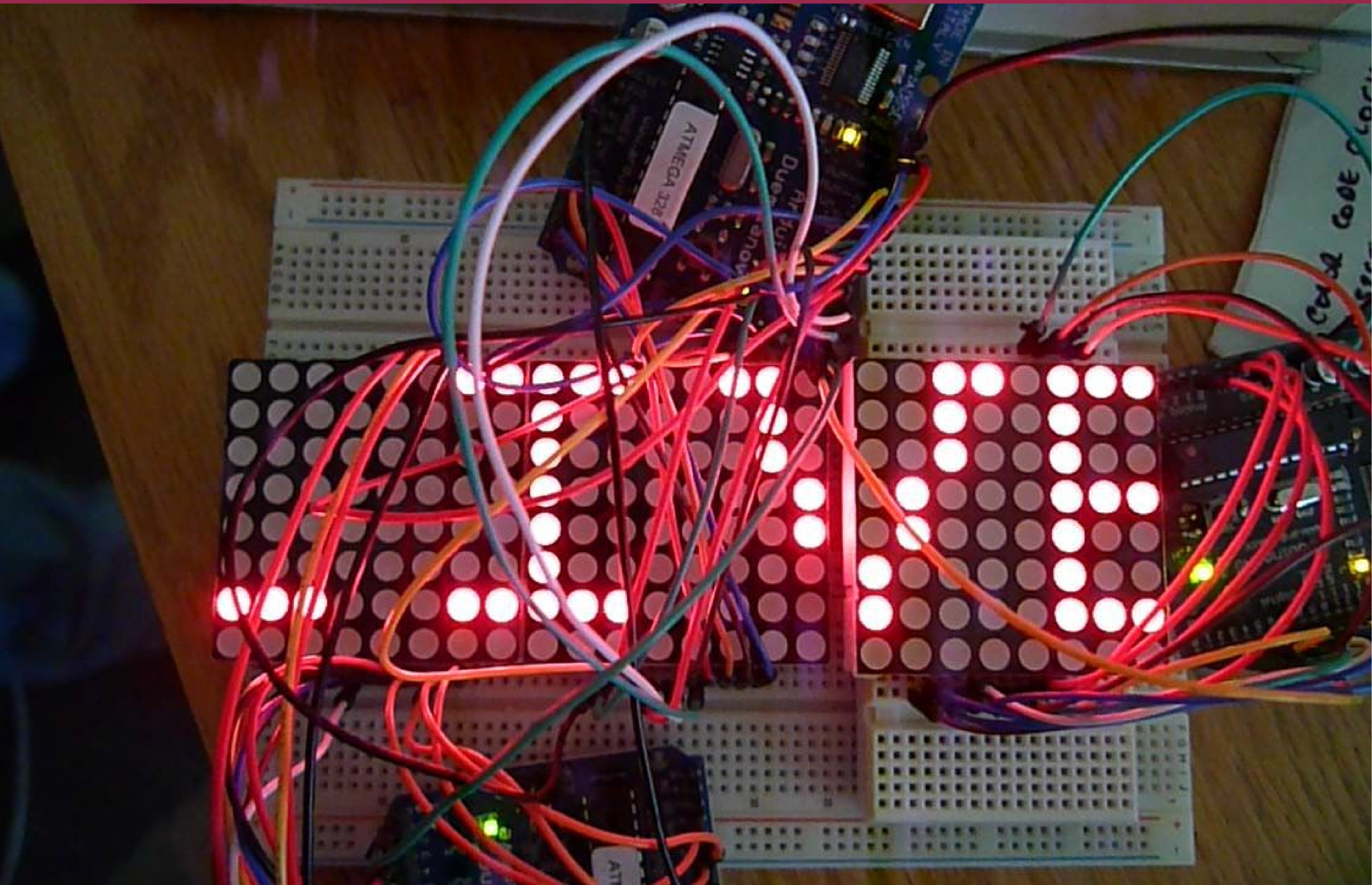
Case study: swarm robotics

- Different from the others: this is looking at *engineering* emergent behaviours, rather than analysing them
  - ... so we enter the process at a different point
- Various applications from the Bristol Robotics Lab, treating robots as “embodied simulations”
- Using e-puck robots, which use AVR and ARM processors and have significant on-board intelligence – no tethering!

The Transterpreter

- Concurrent runtime for embedded devices
- Interpretive VM
- Extremely portable
  - Lego Mindstorms RCX/NXT
  - Assorted robots
  - Atmel AVR (Arduino)
- Used for teaching at Kent and at Allegheny College – undergrad concurrency module, and taster sessions

Distributed, embedded...



The CoSMoS project

- Building a reusable, agile process for constructing and experimenting with complex systems simulations
- We have developed techniques for building scalable, concurrent, distributed simulations
- Plans for the future:
  - Further case studies (e.g. social exclusion)
  - More powerful tools for visualising and analysing simulation results

Parrots demo

Any questions?

Find out more about CoSMoS:
http://www.cosmos-research.org/

Learn concurrent programming with the Arduino:
http://concurrency.cc/