

A Transactional Architecture for Simulation

Tim Hoverd
University of York

Adam Sampson
University of Kent



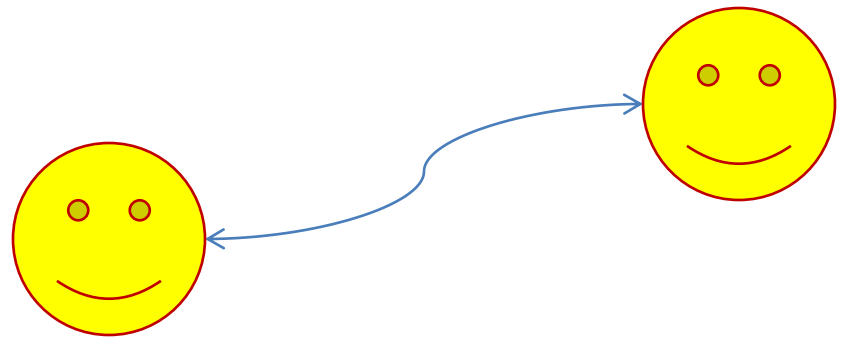
Complex Systems Simulation



- CoSMoS project
 - Building capability in complex systems modelling and simulation across all fields of scientific endeavour
- Complex systems:
 - Large collection of interacting agents
 - In some common environment
 - Often displaying emergent behaviours
- For example:
 - liquid crystals
 - social systems

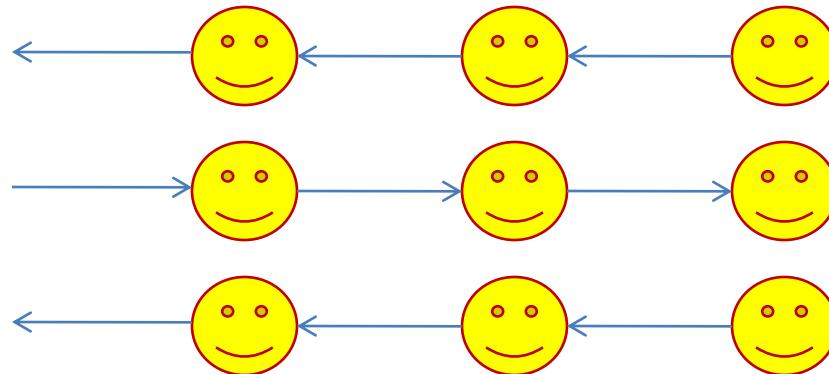
Previous simulations

- Agents talking directly to each other
 - using CSP channels and operation invocation
- Unstructured, unpredictable interactions
- Model checking intractable for large systems



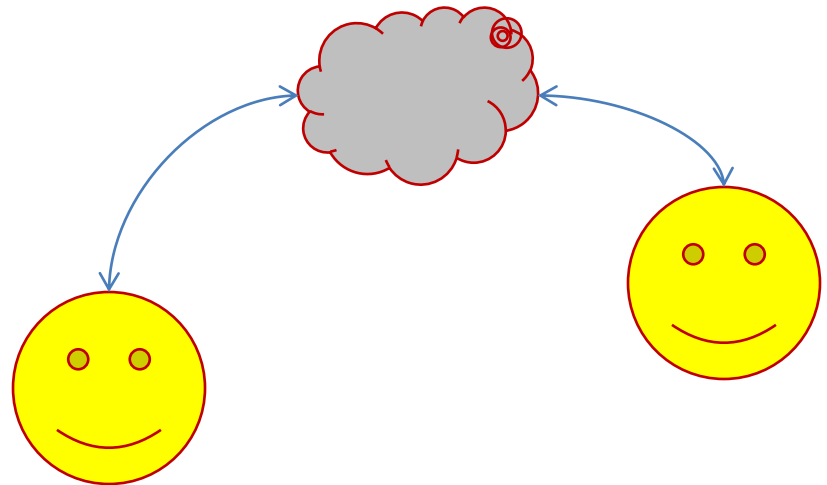
Design rules

- Use specific interaction patterns:
 - proven to be safe
 - e.g. I/O-PAR; agents move in lock-step
- In a large system
 - only allows simple patterns of behaviour



Environment orientation

- Complex systems agents interact only with their environment
- Communications are performed through the environment



What agents do

- Independent cycles of agent behaviour:
 - Retrieve
 - Compute
 - Publish
- Interactions with environment are regular:
 - client/server pattern

A transactional approach

- Agents publish their own state to the environment
- No need for locking:
 - each agent maintains its own state
 - each update must be atomic and durable
- Read-committed isolation
- No rollbacks necessary
 - the real world doesn't roll back
- Implementations:
 - RDBMS, tuplespaces, transactional memory

The rest of the talk

- Embodiment of *environment's* behaviour
- Time and fairness
- Reproducibility
- Robustness

Embodiment

- Environment itself has behaviour
 - Ant trail decay
 - Nutrient uptake
- Can be implemented:
 - as agents
 - or by the environment itself



Time and fairness

- Information propagates at different rates:
 - between different agents
 - between different simulation platforms
- No shared notion of time
 - agents may execute at different rates
 - this does happen in real world complex systems
 - one agent could “run ahead” of all the others

Virtual time

- Give all agents a monotonically increasing notion of “virtual time”
 - need to ensure that agents do not run ahead
 - sloppy synchronisation, allow clocks to drift
 - allows greater parallelism
- The problems of discretising time; continuous time is useful for complex systems simulations

Reproducibility

- Complex systems simulation is said to permit *reproducible* “experiments”.
- These techniques introduce non-determinism
 - updates occur in any order
 - “sloppy” time synchronisation
- Realistic
- Simulation is a scientific instrument

The repeatability tradeoff

- More non-determinism implies a faster simulation
- Another simulation parameter
 - reason about simulations in the same manner as wet experiments
- Debugging
 - reduce time slop
 - re-execution of traces



Robustness

- Complex systems are naturally robust:
 - Immune system works in many situations
 - Rare, but serious, failure modes
- Engineered systems are fragile
 - Engineered complex systems simulations are fragile
- Our approach less fragile
 - agents are more independent
 - communication patterns simpler
- Test for robustness
 - destroy agents and look for emergence to persist

Transactional architecture

- Based on experience with complex systems simulations
- Follows existing high performance architectures
- Experimentation...

