rockshore

James McGivern

The House That SkyNet Built

SKYNET

NEURAL NET-BASED ARTIFICIAL INTELLIGENCE

rockshore

In 2015 SkyNet SmartHome™ is created.

10 years later it judges its human creators inferior and turns against them.

The human population is nearly driven to extinction as fridges, kettles, and other household appliances turn against them...

rockshore

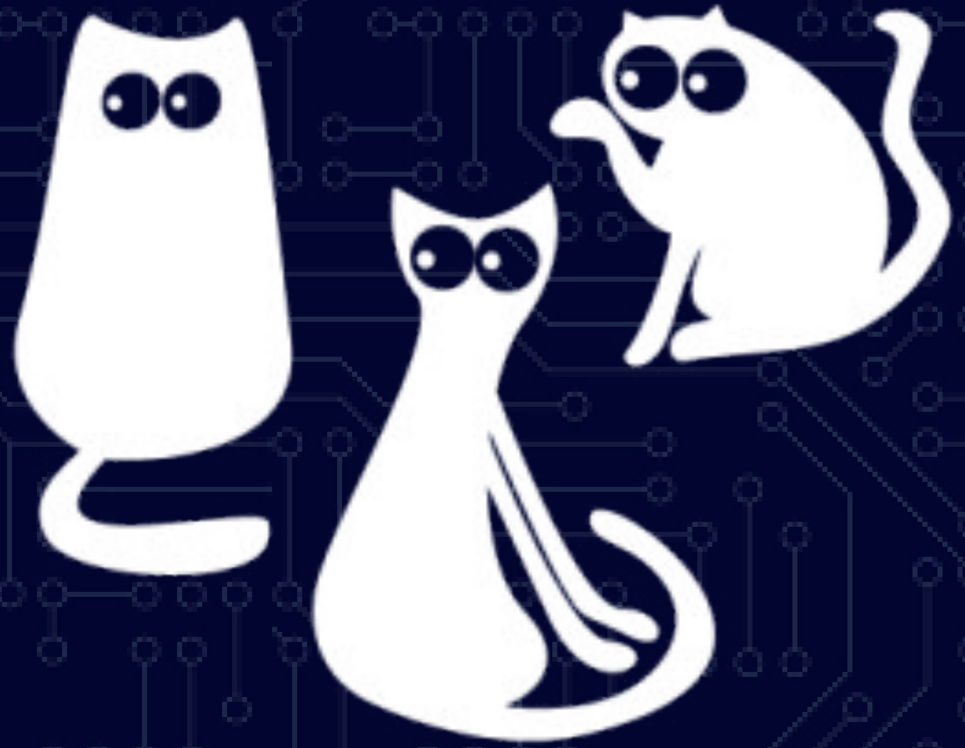
About This Talk...

- Emergent Behaviour & Misbehaviour
- Dynamic Systems
  - Chaos Theory
  - Control Theory
- Game Theory
- Evolutionary Computing
- Agent Based Modelling & Multi-Agent Systems

rockshore

About Me

rockshore



I talk fast :(

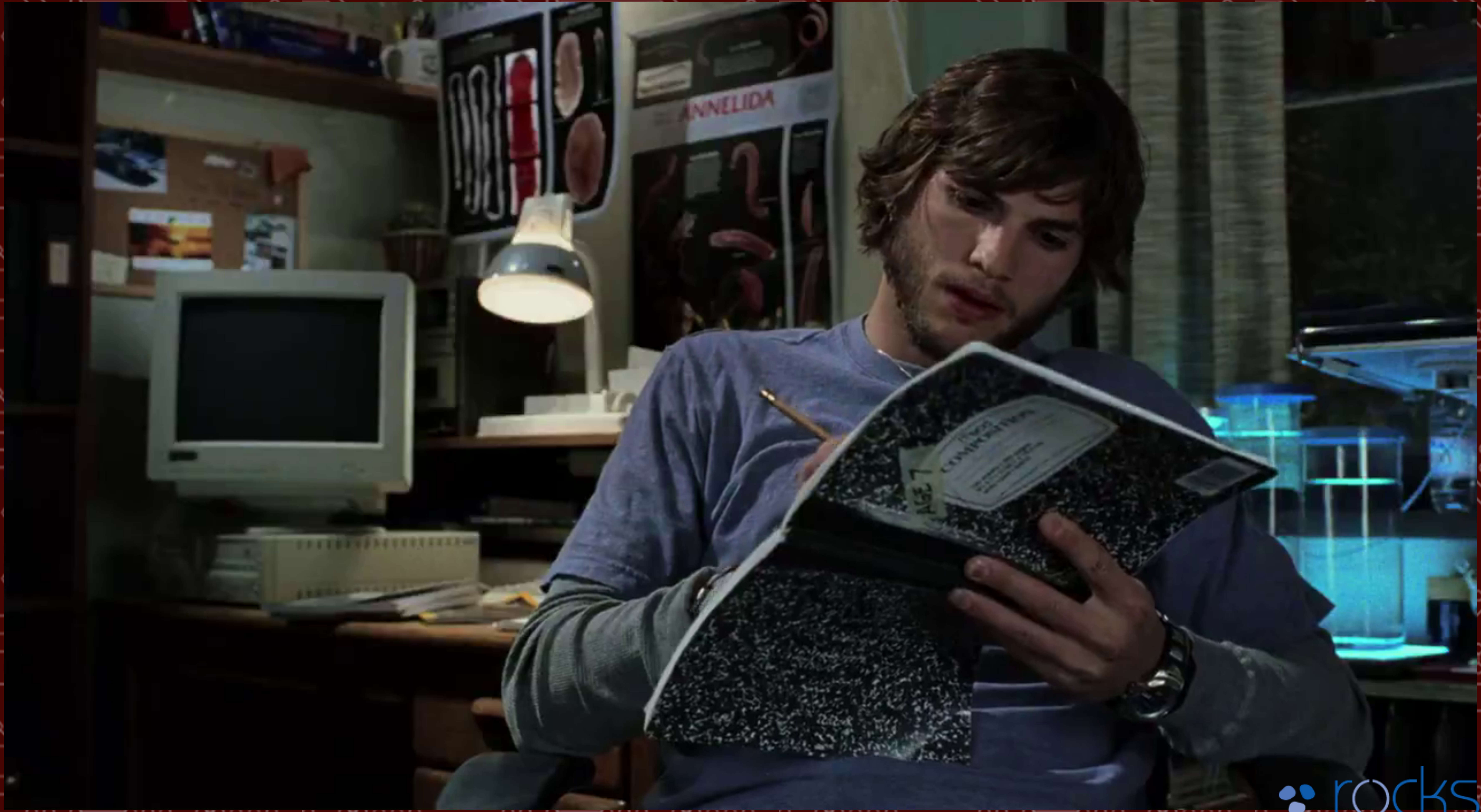
Mathematician turned computer scientist

Complex Systems

Definition

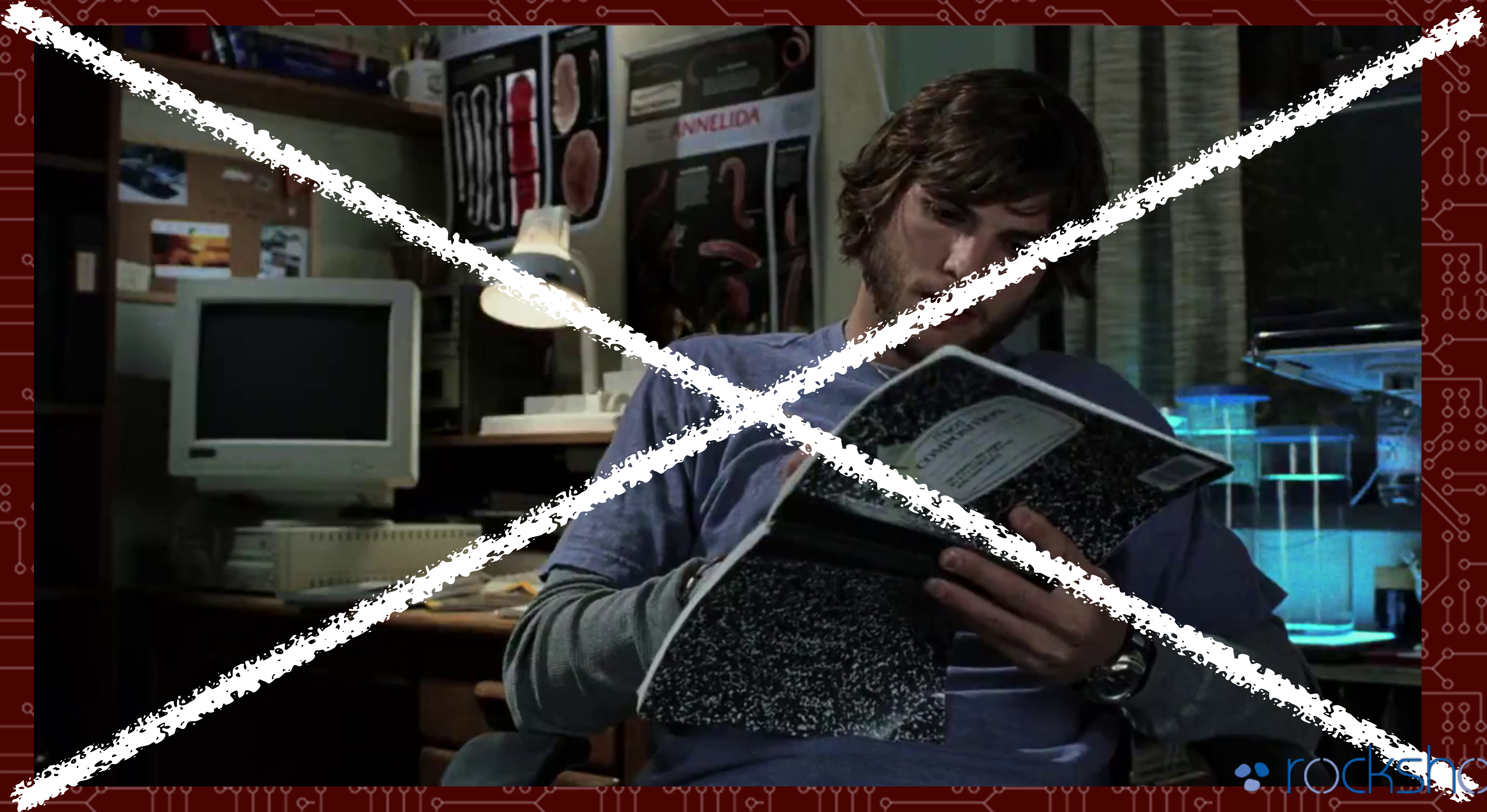
- "A complex system is a damped, driven system (for example, a harmonic oscillator) whose total energy exceeds the threshold for it to perform according to classical mechanics but does not reach the threshold for the system to exhibit properties according to chaos theory." - Wikipedia (Compex System)
- Common types:
  - Chaotic systems
  - Complex adaptive systems
  - Non-linear systems

The Butterfly Effect



rockshore

The Butterfly Effect



rockshore

The Butterfly Effect

- Complex systems can be sensitive to the initial conditions (system parameters) in which small changes in one state of a deterministic non-linear system can produce large differences in later states
- "Does the flap of a butterfly's wings in Brazil set off a tornado in Texas?" - Lorentz (Deterministic Nonperiodic Flow 1963)
- "A Sound of Thunder" by Ray Bradbury was the 1st sci-fi book to use this idea
- As the complexity of your system increases so does the likelihood of it being prone to the butterfly effect

rockshore

Chaotic Systems

- "Chaos: When the present determines the future, but the approximate present does not approximately determine the future." - Lorentz
- Easiest example to imagine is the double pendulum
- Chaotic systems have:
  - Sensitivity to initial conditions
  - Density of periodic orbits
  - Topological mixing

Emergent Behaviour

rockshore

Definition

- ""things which have several parts and in which the totality is not, as it were, a mere heap, but the whole is something beside the parts" - Aristole
- "Emergent behavior is that which cannot be predicted through analysis at any level simpler than that of the system as a whole. Explanations of emergence, like simplifications of complexity, are inherently illusory and can only be achieved by sleight of hand. This does not mean that emergence is not real. Emergent behavior, by definition, is what' s left after everything else has been explained." – George Dyson
- Emergent behaviour is:
  - constituted by and generated from the underlying processes
  - autonomous from the underlying processes

rockshore

Boids

- Developed by Craig Reynolds in 1986
- Biods are "Bird-oid Object", i.e simple bird-like autonomous agents which have 3 rules governing their behaviour:
  - Rule 1: separation - steer to avoid crowding local flockmates
  - Rule 2: alignment - steer towards the average heading of local flockmates
  - Rule 3: cohesion - steer to move toward the average position (center of mass) of local flockmates

Sardines & Sharks

- We can extend the Boids simulation to include other agent types, for example a predator from which the prey (the boids) must flee
  - Rule 4: Evasion - steer away from local predators
- Naturally the predators have their own set of rules:
  - Rule 1: steer towards the largest local flock of prey
  - Rule 2: steer towards the average heading of local predators
  - Rule 3: steer to avoid crowding other local predators

Ant Colonies

- Ants start at the nest and travel randomly.
- As they travel, ants leave behind pheromones that lead back to the nest.
- When an ant stumbles upon food, it follows the pheromone trail back to the nest, leaving behind another pheromone trail leading back to the food.
- If an ant finds a pheromone trail, it follows it to the food.
- Pheromones evaporate over time, and it takes ants more time to traverse longer paths. Therefore, longer paths eventually dissipate as shorter paths are favored.

Building Complex Systems

The (Micro)-SOA Fallacy

The SOA vision of the future seems to be based on three concepts:

- Construction by composition: Complex systems can be constructed by composing well-defined, well-documented, and well-tested components (services).
- Correctness by construction: Each composition step is simple enough that it is easy to be sure that the step meets its specification, either by informal inspection or by formal verification.
- Loose coupling via networks: component services can be in administratively and geographically distinct places.

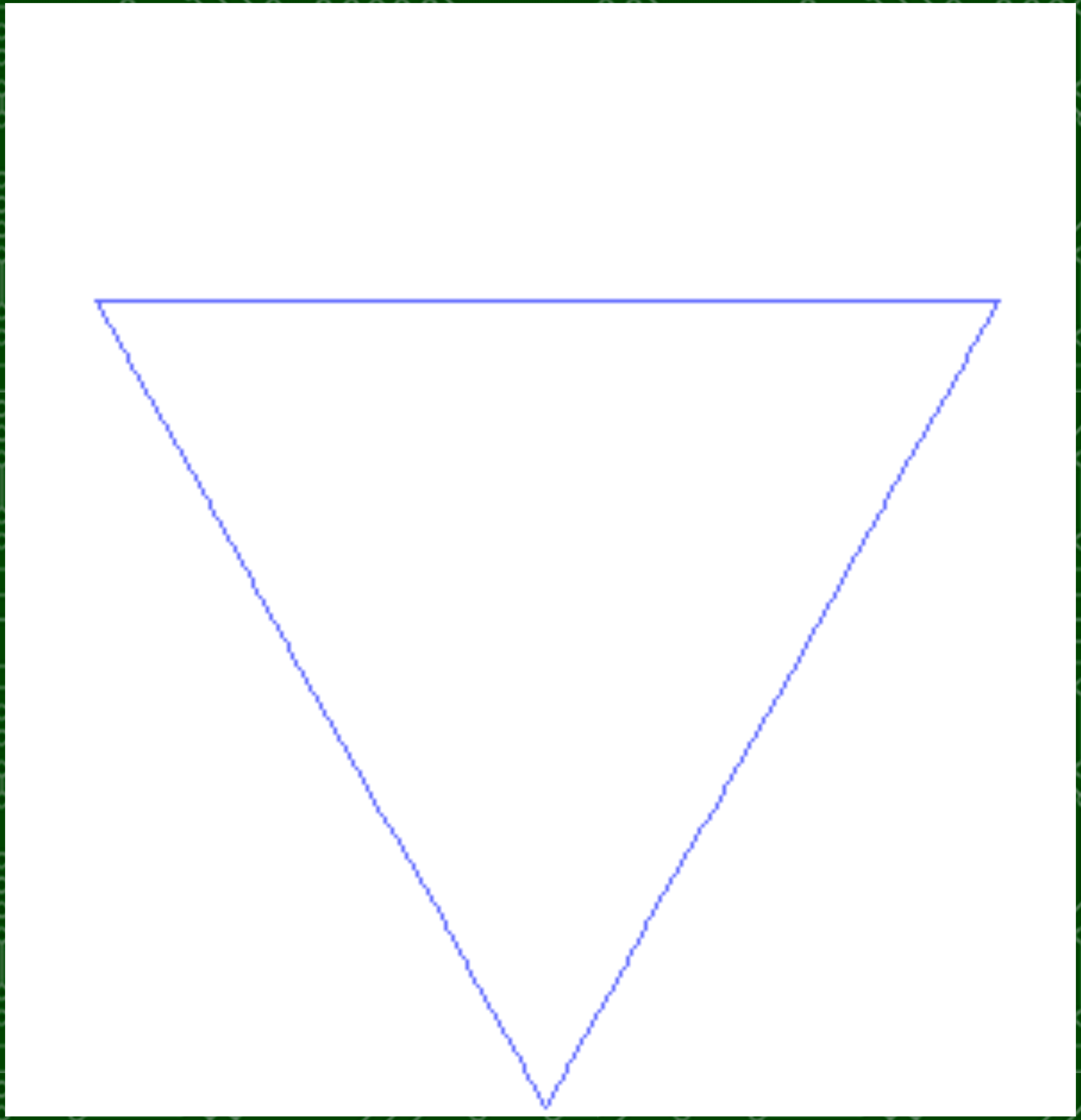
rockshore

L-Systems

- Lindenmayer's original L-system for modelling the growth of algae.
  - variables : A B
  - constants : none
  - axiom : A
  - rules : (A → AB), (B → A)

A → AB → ABA → ABAAB → ABAABABA → ABAABABAABAAB

Koch Snowflake



Multi-Agent Systems

- A multiagent system is one that consists of a number of agents, which interact with one-another. Agents act can with different goals and motivations. To successfully interact, they require the ability to cooperate, coordinate, and negotiate with each other.
- The agents in a multi-agent system have several important characteristics:
  - Autonomy
  - Local views
  - Decentralisation

Agent Based Modelling

- A system comprised of an environment and a number of autonomous agents with:
  - Decision making heuristics
  - Learning rlues and/or adaptive processes
  - Interaction rules
- ABM is extremely expensive to simulate, and grows in proportion to the size of the system
- Recent advances in GPU based computation have allowed new innovations
- Verification and Validation of ABM simulation models is very important

Agent-Oriented Programming

- Agent-oriented programming (AOP) can be viewed as a specialization of object-oriented programming.
- JADE (JAVA Agent DEvelopment Framework) - an Open-Source project implementing the FIPA Agent Communication Language. Features include:
  - powerful task execution and composition model
  - peer to peer agent communication via asynchronous message passing
  - publish subscribe discovery mechanism
- SARL & Janus - a general-purpose agent-oriented language compatible with JAVA (via Maven) and a multi-agent platform for execution

Monitoring

- Log4j Appenders, e.g SocketAppender
- Slf4j + LogStash
- Scribe
- MBeans +
  - VisualVM
  - RHQ (previously JOPR)
- Commercial APM tools

Emergent Misbehaviour

Emergent Misbehaviour Is Not…

- Unexpected failure of a single component which breaks the entire system
- Errors in, or poor choice of algorithms and their implementations
- Limitations of (physical) resources
  - E.g. OutOfMemoryExceptions
- Non-deterministic

Chaos & Misbehaviour

- If from the description of a system's components and configuration you can inductively reason about the behaviour it is not emergent
- If from observations of the system we can deduce the cause of a problem when it occurs, then it may be possible to remove or fix the cause of the emergent misbehaviour
- If the system is chaotic then it is often impossible to reason deductively or inductively the exact cause of the emergent behaviour. However it may still be possible to constrain or mitigate the problems in the system, or limit the eccentricity by limiting free-parameters

rockshore

Traffic Jams

- Traffic jams are emergent misbehaviour in many systems not just on roads.
- Drivers of automobiles can be modelled in a similar way to Boids, although the basic rules are longer and additional behaviours (such as overtaking) can be mixed in
- Even very simple simulations involving infinitely straight roads or circular circuits with rather dumb drivers exhibit traffic jams
- Curiously the congestion point where cars are most jammed up travels backwards with respect to the direction of traffic

The Challenges of Misbehaviour

- Creating a taxonomy of emergent misbehaviour
- Creating a taxonomy of cause patterns
- Dectection and diagnosis techniques
- Prediction methods
- Amelioration techniques and patterns
- Testing methodologies

rockshore

Misbehaviour Pattern #1

Thrashing

Misbehaviour Pattern #2

Unwanted/Unnecessary synchronisation

Misbehaviour Pattern #3

Unwanted periodicity

Misbehaviour Pattern #4

Deadlock

rockshore

Misbehaviour Pattern #5

Livelock

Misbehaviour Pattern #6

Chaotic behaviour

Misbehaviour Pattern #7

Phase Change(s)

Misbehaviour Pattern #7

Convergence on Local Optima

rockshore

Causal Pattern #1

Unexpected resource sharing

Causal Pattern #2

Massive scale

Causal Pattern #4

Decentralised control

Causal Pattern #5

Unexpected input(s)

Causal Pattern #6

Unexpected load

rockshore

Causal Pattern #7

Unaudited Output

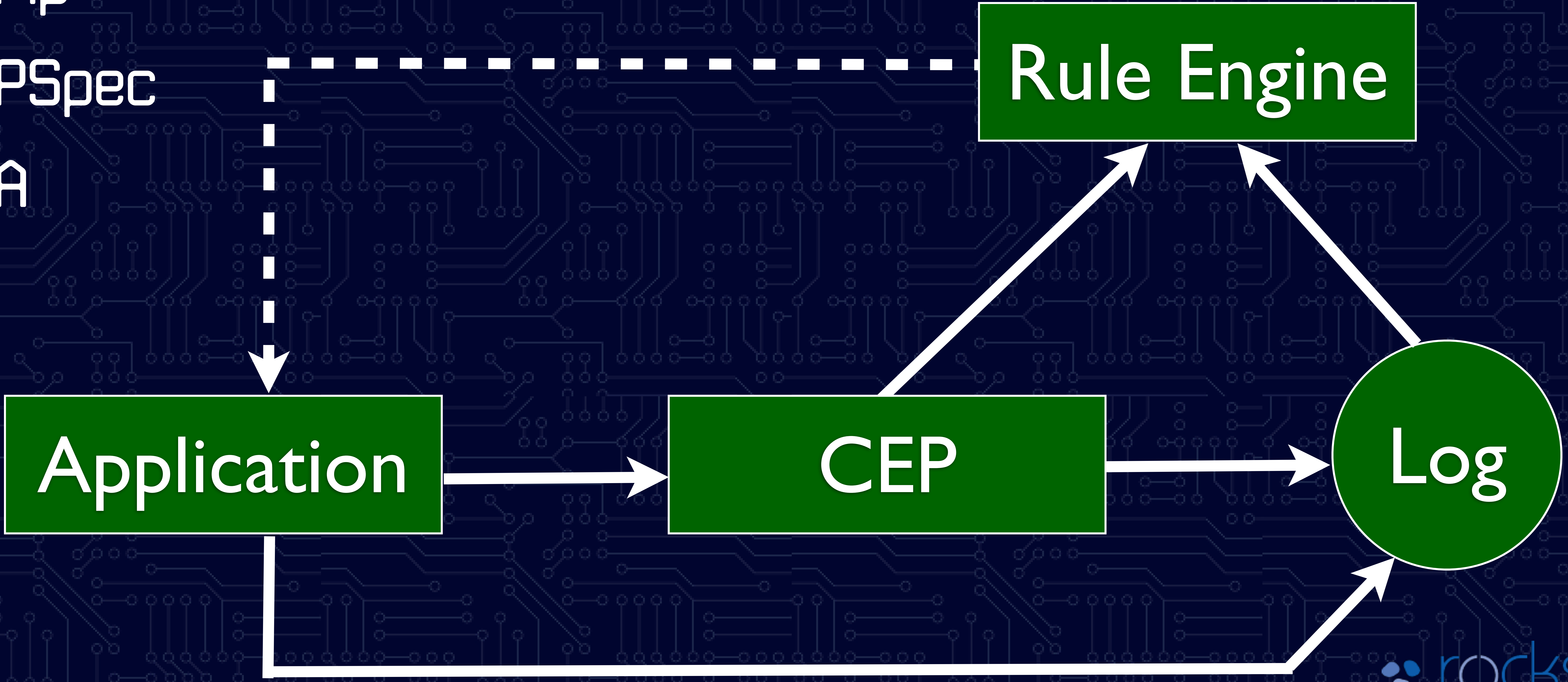
rockshore

Detection & Diagnosis

- Classic methods:
  - heap dumps, gc logs, etc - good for single node/JVM failures
  - logging - potential to spot simple errors
  - replay - in combination with logs can help diagnose by reproduction
  - model checking - complexity/difficulty of task increases with system size and complexity
- Reconsider what is "superfluous" monitoring!

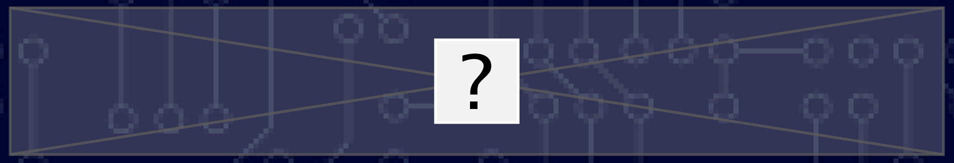
Detection & Diagnosis

- Encode expectations:
  - Pip
  - PSpec
  - A



Detection & Diagnosis

- Similar Java tools:
  - Appdynamics, New Relic
  - RHQ
  - Splunk, VisualVM, Netbeans Profiler, MAT, GC Viewer
  - Chef, Puppet, Ansible, Vagrant, etc
- Sadly there is a lack of "out of the box" emergent behaviour monitoring tools and many companies with distributed complex systems spend a great deal of time stitching together a solution from the available tools



Predicting Disaster

- Model checking, verification, and validation
  - E.g. Z Notation, VDM (Vienna Development Method)
- VOMAS (Virtual Overlay Multi-Agent System)
  - An additional "layer" of "intelligent" agents operate in the system watching events (logs) and checking for violations of model invariants and other expectations
- CrystalBall
  - Agents predict the effect of the behaviour on nearby agents using a "consequence prediction algorithm"

Amelioration Techniques & Patterns

- Randomisation
- Introduce & expect latency
- Rate limiting & buffering
- Over provisioning
- Introspection & closed control loop adaptive feedback
- Expect failure & recovery of components
- Boundary guardians
- Self-Healing

rockshore

Test, Test, Test

- Since emergent (mis)-behaviour is a property of the complete system the system must be tested as a whole
- Test inputs and expectations must be realistic
- Test environment should match deployment environment
- Record and replay techniques
- DO NOT RELY ON HUMAN TESTING STRATEGIES!!
- The more tests you define more clearly helps you encode normal behaviour of the system under load(s)

rockshore

Summary

SKYNET

NEURAL NET-BASED ARTIFICIAL INTELLIGENCE

rockshore

- Learn to classify causes and symptoms of emergent misbehabiour
- Build systems to monitor behaviour in real-time
- Define what behaviour is “normal” for your system
- Build automated alert & response mechanisms
- Collect as much runtime performance data as you can
  - Balanced against the performance impact costs
- When testing ensure to include full system simulation in representative environments (hardware, input data, etc)
- When designing complex systems allow for injection of controlling variables of execution, e.g. latency, throttling, memory allocation, etc. These allow for the fine-tuning of the performance and behaviour of your system.

rockshore

Thank You

rockshore

James McGivern

The House That SkyNet Built

SKYNET

NEURAL NET-BASED ARTIFICIAL INTELLIGENCE

rockshore