Self-Managing Systems: a bird’s eye view

Márk Jelasity
Outline

- Background
  - Historical perspective
  - Current state of IT
- What do we need?
  - Desired self-* properties
  - The human factor
- How do we get there?
  - Autonomic computing
  - Grassroots self-management
- Course outline
XIX century technology

- Mechanical Clocks and Sewing machines
  - Long 40 page manuals of usage
  - Two generations to become widely used
- Phonograph
  - Edison’s version unusable (geeky)
  - Berliner: simplified usage, became ubiquitous
XIX century technology

- Car
  - 1900s: “mostly burden and challenge” (Joe Corn)
    - Manual oil transmission, adjusting spark plug, etc,
    - Skills of a mechanic for frequent breakdown
    - Chauffeur needed to operate
  - 1930s: becomes usable
    - Infrastructure: road network, gas stations
    - Interface greatly simplified, more reliable
XIX century technology

Electricity

- Early XXth century
  - Households and firms have own generators
  - “vice president of electricity” (like now: “chief information officer”)

- One generation later
  - power grid: simplified, ubiquitous power plug, no personnel
Originally, all kinds of technology needs lots of human involvement

- New inventions are typically “geeky”, need expertise to install and maintain
- In general, the “default” seems to be human work, due to its flexibility and adaptivity: in an early stage it is always superior to alternatives
Usual path of technology

- Eventually, humans are removed completely or mostly by the technology becoming simple (for humans) and standardized
  - To increase adoption and sales (electricity, cars, etc)
  - To decrease cost (industrial revolution, agriculture)
  - To allow super-human performance (space aviation)
- Simplicity of usage often means increased overall systems complexity (is this a rule?)
IT now

“IT is in a state that we should be ashamed of: it’s embarrassing”
Greg Papadopoulos, chief technologist, Sun

- IT project failure or delay
  - 66% due to complexity, 98% for largest projects (over $10m)
- IT spending
  - 15 years ago: 75% new hardware 25% fixing existing systems
  - Now: 70-80% fixing and maintaining existing systems
Example systems

- Personal computer
  - Hardware, software components
  - Small scale, single owner, single user
- In-house data-center
  - Collection of servers
  - Middle scale (10-10000), single owner, central control, many users (applications) with more or less common interest (cooperation)
Example systems

- E-sourcing provider (ASP, SSP, cycle provider)
  - Storage, compute, etc services
  - Middle scale (thousands of servers)
  - Single owner, central control
  - Many users, with different (competing) interests
  - Governed by QoS agreements
Example systems

- Supply chain (supply network)
  - Thousands of outlets, suppliers, warehouses, etc
  - Can be global and large scale (Walmart) with many participants
  - Participants are selfish and independent (maximise own profit)
  - Can be decentralized, no central decision making
Example systems

- P2P
  - Simple computing and storage services
  - Very large scale
  - Fully decentralized
  - Participants are individuals
  - Interests of participants ?? (motivation to participate, etc)
  - non-profit, non-critical apps
Example systems

- Grid
  - Compute, storage, etc resources
  - Can be very large scale
  - Decentralized (?), dynamic
  - Well designed and overthought sharing
  - Complex control
    - Virtual organizations (consisting of ASPs, SSPs, individuals, academy, etc)
    - Policies based on virtual organizations
Problem statement

- Information systems are very complex for humans and costly to install and maintain
- This is a major obstacle of progress
  - In industry
    - IT costs are becoming prohibitive, no new systems, only maintenance
    - Merging systems is extremely difficult
  - For ordinary people
    - electronic gadgets, computers, etc, cause frustration, and discomfort, which hinders adoption
  - Cutting-edge IT (research and engineering)
    - scalability and interoperability problems: human is the “weakest link” in the way of progress
What do we need?
What do we need?

- We need self-managing information systems
- Industry and academy are both working towards this goal
  - IBM: autonomic computing
  - Microsoft: dynamic systems initiative
  - HP: adaptive enterprise
  - Web services
  - Grid services
  - Pervasive computing
What does self-management involve?

- We use IBM’s autonomic computing framework to define basic requirements
  - High level, user friendly control
  - Self-configuration
  - Self-healing
  - Self-optimization
  - Self-protection
Self-configuration

“real plug-and-play”
- A component (software service, a computer, etc) is given high level instructions (“join data-center X”, “join application Y”)

Application configuration (self-assembly)
- Applications are defined as abstract entities (a set of services with certain relationships)
- When started, an application collects the components and assembles itself
- New components join in the same way

[Self-assembly, self-organization]
Self-optimization is about making sure a system not only runs but its optimal.

- All components must be optimal.
- The system as a whole must be optimal.
- These two can conflict.
- There can be conflicting interests: multi-criteria optimization.

[Self-adaptation]
Self-healing, self-protection

- **Self-healing**
  - System components must be self-healing (reliable, dependable, robust, etc)
  - The system as a whole must be self-healing (tolerate failing components, incorrect state, etc)
  - [self-stabilizing, self-repair]

- **Self-protection**
  - Malicious attacks: DOS, worms, etc
Easier or more Difficult?

- Only rare high level interaction?
  - People get bored and have to face problems “cold” (aviation)
  - When there is a problem, it is very difficult and needs immediate understanding
  - Solution in civil aviation: machines help humans and not vice versa (really?). But: in space aviation, machines are in charge

- Lack of control over small details and so lack of trust?
  - IBM: we’ll get used to it gradually. (Maybe actually true.)
Some confusion

“Usable autonomic computing systems: the administrator’s perspective” (ICAC’04) (authors from IBM)

The paper is about how admins will do what they do now in the new framework

That’s the whole point

It’s like saying “usable usable computing systems”
How do we get there?
How do we get there?

- General consensus: open standards are essential (as opposed to MS)
- Two approaches
  - Self-awareness: simplicity through complexity
    - Self-model (reflection)
    - Environment model
    - Planning, reasoning, control (GOFAI)
  - Self-organization: simplicity through simplicity
    - Emergent functions through very simple cooperative behavior (biological, social metaphors)
- These two can compete with or complement each other
Autonomic computing architecture: a self-aware approach

- Autonomic elements
- Interaction between autonomic elements
- Building an autonomic system
- Design patterns to achieve self-management
Self-managing element

- **Must**
  - Be self-managing
  - Be able to maintain relationships with other elements
  - Meet its obligations (agreements, policies)

- **Should**
  - Be reasonable…
  - Have several performance levels to allow optimization
  - Be able to identify on its own what services it needs to fulfill its obligations
Self-managing element

- Policies
  - Action policies
    - If then rules
  - Goal policies
    - Requires self-model, planning, conceptual knowledge representation
  - Utility function policies
    - Numerical characterization of state
    - Needs methods to carry out actions to optimize utility (difficult)
Interaction between elements

- Interfaces for
  - Monitoring and testing
  - Lifecycle
  - Policy
  - Negotiation, binding
- Relationship as an entity with a lifecycle
- Must not communicate out-of-band, only through standard interfaces
Special autonomic elements for system functions

- Registry
  - Meeting point for elements
- Sentinel
  - Provides monitoring service
- Aggregator
  - Combines other services to provide improved service
- Broker, negotiator
  - Help creating complex relationships
Design patterns for self-configuration

- Registry based approach
  - Submit query to registry
  - Build relationship with one of the returned elements
  - Register relationship in registry
- In general: discovery
  - Service oriented paradigm, ontologies
- Longer term ambition: fully decentralized self-assembly
Design patterns for self-healing

- Self-healing elements: idiosyncratic
- Architectural self-healing
  - Monitor relationships and if fails, try to replace it
  - Can maintain a standby service to avoid delay when switching
  - Self-regenerating cluster (to provide a single service) where state is replicated
Design patterns for self-optimization and self-protection

- **Self-optimization**
  - Market mechanisms
  - Resource arbiter (utility optimization)

- **Self-protection**
  - Self-healing mechanisms work here too
  - Policies
A sidenote on the name

- Autonomic computing is bio-inspired: autonomic nervous system: maintains blood pressure, adjusts heart rate, etc, without involving consciousness

- [disclaimer: I’m not a biologist…] the ANS
  - Is based on a control loop, central control by specific parts of the brain (hypothalamus, sympathetic and parasympathetic systems)
  - However, no reflection, self-model and environment model (???)
  - Many functions, such as healing and regeneration are fully decentralized (no connection to central nervous system) (???)
Advantages of self-awareness

- Explicit knowledge representation: potentially more "intelligent"
  - Better in semantically rich and diverse environments
  - Plan and anticipate complex events (prediction)
- Possibility to reason about and explain own behavior and state
  - More accessible administration interface
  - Higher level of trust from users
- Incremental
Issues with self-aware approaches

- In large and complex systems emergent behaviour is inevitable, even if centrally controlled in principle (parasitic emergence)
  - Complex networks (scale free)
  - Supply chains
    - Chaotic, unpredictable behavior even for simple settings
  - Cooperative learning: often no convergence
Issues with self-aware approaches

- Large systems with no single supervisor organization
  - Decentralized by nature so the only way is a form of self-organization (market-, bio-inspired, etc)
  - Grid: multiple virtual organizations
  - P2P: millions of independent users
  - Supply chain (network): independent participants
Issues with self-aware approaches

- Many critical components
  - Esp. high level control components
  - Less resilient to directed attacks
  - Potential performance bottlenecks

- Hugely ambitious
  - Controlled systems like airplanes are not like information systems (hint: we still don’t have automated cars: it’s more like the IT problem)
  - Needs to solve the AI problem in the most general case, like in the car automation problem, although can be done gradually
Issues with self-aware approaches

- Simplicity means extremely increased complexity behind the interface
  - Cars, power grid: hugely complex, extremely simple interface (early cars were much simpler)
- Implementation is more expensive
No generic architecture proposal yet.
- Is it possible? maybe
- Does it make sense? certainly

Some attempts have been made here (Bologna)
- Highly self-healing and self-optimizing system services:
  - Connectivity (lowest layer)
  - Monitoring (aggregation)
  - Self-assembly (topology management)
- Could be added (among other things)
  - Application service discovery, application self-assembly
- Can be combined with self-aware architecture
Advantages of self-organization

- Extremely simple implementation (no increased complexity): lightweight
- Potentially extremely scalable and robust: self-healing, self-optimization, etc for free
- Works in hostile environments (dynamism, across administration domains, etc)
Issues with self-organizing approaches

- Reverse (design) problem is difficult (from global to local)
  - Local behavior can be evolved (evolutionary computing)
  - Design patterns for building services, and interfaced in a traditional way
- Trust of users seems to be lower
- Control is very difficult (and has not been studied very much)
- Revolutionary (not incremental)
Relationship of self-organization and self-awareness

- Since in large complex systems there is always emergence, it is always essential to understand (perhaps unwanted) self-organization.
- Especially in large-scale, dynamic settings self-organization is always an alternative to be considered.
- Many applications already exist based on emergence, most notably in P2P, that are increasingly attractive for the GRID and other autonomic systems.
- A mixed architecture is also possible.
Course outline
Basic approach behind the structure of the course

- Autonomic comp., P2P comp., distributed comp., middleware, GRID, Web, complex systems, agent based comp., planning, semantic web, machine learning, control theory, game theory, AI, global optimization etc.

- In spite of this huge effort, and many relevant fields, everything is still in motion

- Idea is to pick the key topics that
  - stand out as promising and relevant
  - possibly span many fields
  - are suitable to fill the bird’s eye view with detail (that is, we mostly use this introduction as a skeleton)
High level user control

- Motivation
  - A common theme is way of allowing high level control to ease the burden on users and admins

- Outline
  - Policy types in self-aware systems (rule, goal (planning), utility (optimization))
  - Control (and the lack of it) in self-organizing systems
Self-configuration

Motivation

- Another common theme is the study of ways a complex system can self-assemble itself

Outline

- Self-configuration in service oriented systems (eg GRID)
- Self-assembly in self-organizing systems (P2P (T-Man), mobile robots, etc)
Learnign and adaptive control

Motivation

One popular way of self-optimization is modeling systems through learning, and applying adaptive control techniques.

Outline

- Basic concepts in adaptive control
- Application of control in information systems
- Some machine learnign techniques
- Application of learning in modeling, optimizing and controlling systems
Recovery oriented computing

- Motivation
  - A prominent and popular direction for self-healing in complex systems is adaptive (micro-) reboot and rejuvenation

- Outline
  - The Cornell-Berkeley ROC project
  - Other results related to restart and rejuvenation
Game theory, cooperation

- Motivation
  - In decentralized systems involving independent agents, negotiation, bidding, market-inspired techniques are often used. Besides, studies of the emergence cooperation are highly relevant.

- Outline
  - Self-optimization through utility optimization with market-inspired techniques
  - Emergence of cooperation: getting rid of the tragedy of the commons
Reinforcement learning

Motivation
- Reinforcement learning (Q-learning) is a widely used non-supervised technique for adaptive self-optimization in a large number of fully distributed environments

Outline
- Introduction to reinforcement learning
- Ants
- Distributed Q-learning
Complex networks

- Motivation
  - As an outstanding illustration of parasitic emergence in large complex systems and its crucial effects on performance and robustness of information systems

- Outline
  - Basic concepts (random, scale-free, small world networks)
  - Effect on robustness (self-protection capability)
Motivation

- A major representative of already successful fully distributed self-organising approaches is the class of gossip-based protocols.

Outline

- Intro to gossiping
- The Astrolab environment (self-healing, monitoring, etc)
- Other gossip based approaches (self-healing with newscast, etc)
Wild stuff

- Motivation
  - Just to relax during the last lecture…
- Outline
  - Invisible paint, reaction-diffusion computing, swarm spacecraft and other goodies…
Some refs

- Most important papers this presentation was inspired by or referred to

- The course website
  - [http://www.cs.unibo.it/~jelasity/selfstar05.html](http://www.cs.unibo.it/~jelasity/selfstar05.html)