From autonomic computing to autonomic ICT

Fabrice Saffre
Pervasive ICT Research Centre

Autonomic principles

- The trend towards self-configuration, self-protection etc. championed by IBM is widely referred to as “autonomic computing”.
- Though finding its origin in “pure” research (biologically inspired systems), it has gained so much momentum and widespread endorsement that many implementations now exist.
- However, they are mostly “node-centric” as opposed to “network-centric”.
- This creates the perfect conditions for complex system behaviour (many interacting units making autonomous/selfish decisions on the basis of locally available information).

Expected benefits

- More agile computing assets, capable of responding adaptively to unique, changing and unpredictable user demands.
- More robust software, capable of self-diagnostic and of actively and autonomously seeking to avoid “unsafe” configurations.
- Reduced cost of ownership (i.e. a direct and highly desirable consequence of increased robustness).
- Reduced “downtime” (ibid).

Fabrice Saffre, 2005

The origins of complexity

- Networks are becoming so dynamic and complicated that the only viable management option is to *make* them complex…
- Because we have no choice but to gradually switch from centralised to decentralised control, we are effectively sowing the seeds of complexity.
- We must learn to live with and take advantage of the emergent properties arising from the interaction of many system constituents, not try to counter them.

BT

Self-organisation

- *By definition*, a system composed of units making autonomous decisions based on locally available information can only be "driven" to a desirable state by leveraging self-organisation.
- This requires engineering the reasoning and decision-making engine running on individual units so as to promote the emergence of the "right" collective behaviour.
- In turn, this means adapting the predictive techniques of *natural* complexity science (both analytical and numerical) to meet the needs of *artificial* complex systems designers.
- It seems relatively trivial *in principle*, but experimental validation requires prototype implementation, which is a serious issue.

Practical applications

- Autonomic service deployment: module-based applications could greatly benefit from “on-the-fly” adjustment to unpredictable usage patterns (i.e. “who needs what service, where and when?”).
- Autonomic resources accounting and allocation: “on-demand” distributed computing (i.e. seamless Grid) requires real-time balancing of the offer and demand, which could be achieved via unsupervised negotiation between potential collaborators.
- Self-organising ad-hoc networks: social differentiation (specialisation) and/or decentralised radio spectrum management (e.g. via cross-inhibition) can enhance usability and/or longevity.

Conclusion

- The borders between:
  - Autonomic computing/communication
  - Networking and services
  - Pervasive computing
  - Resource sharing
  - Software design and engineering

  are fading rapidly…
- Because they all share the same problem: less control, increased complexity, poor understanding of artificial systems' emergent properties.

Conclusion (2)

- The corresponding industries have increasingly overlapping markets with, e.g., Telcos and IT companies now competing to provide ICT services.
- The race is on, and whoever can demonstrate that they've "cracked the complexity nut" *in practice* will have a decisive advantage.
- Because they will be able to offer *new* and *cheap* ICT solutions that are *predictably* and *reliably* efficient in the unpredictable and unreliable world of the "network economy".