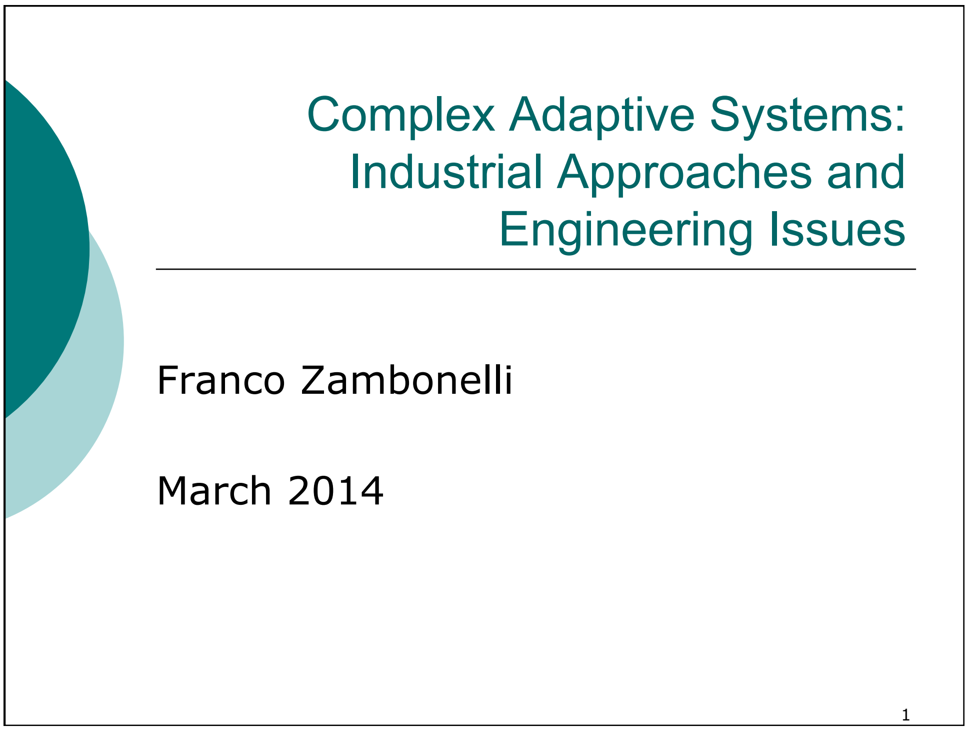
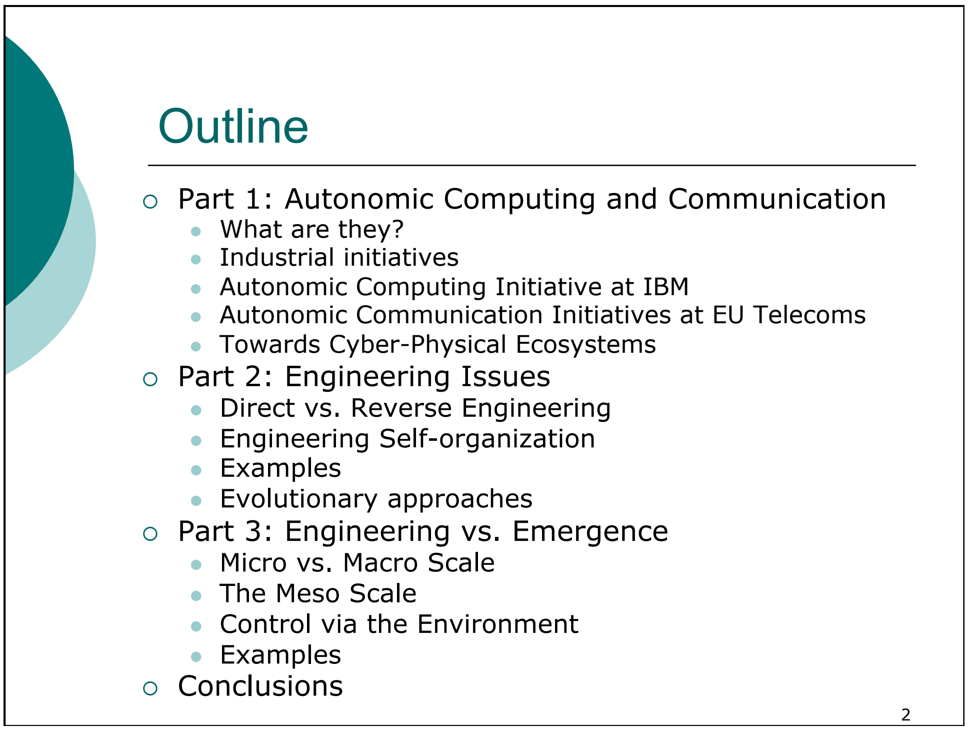


Complex Adaptive Systems: Industrial Approaches and Engineering Issues

Franco Zambonelli

March 2014



Outline

- Part 1: Autonomic Computing and Communication
  - What are they?
  - Industrial initiatives
  - Autonomic Computing Initiative at IBM
  - Autonomic Communication Initiatives at EU Telecoms
  - Towards Cyber-Physical Ecosystems
- Part 2: Engineering Issues
  - Direct vs. Reverse Engineering
  - Engineering Self-organization
  - Examples
  - Evolutionary approaches
- Part 3: Engineering vs. Emergence
  - Micro vs. Macro Scale
  - The Meso Scale
  - Control via the Environment
  - Examples
- Conclusions

Part 1

- Autonomic Computing and Communication

What are they?

- Industry-driven research initiatives
- “Autonomic”
  - The term is borrowed from the “autonomic nervous system”
- Related to the idea of:
  - Giving modern ICT systems a sort of “nervous system”
  - Capable of reacting to contingencies and of regulating in autonomy the overall metabolism of such systems
  - Metabolism = Functional and Non-functional behaviours
  - i.e., self-management, self-adaptation, self-organization, self-healing, self-configuration, etc. “self-*” features
- A first attempt at learning form the lessons of complex adaptive system towards the production of better ICT systems, network, and software
  - We will see how agents, multiagent systems, swarm systems, etc. play a central inspiring role

Autonomic Computing vs Autonomic Communication

- Two different perspectives on trying to embed self-* features in modern ICT systems
- Autonomic Computing:
  - Focus on resource management and reliability (large data- and service centres, large service systems)
  - Main drivers: IBM, Intel, HP
- Autonomic Communication:
  - Focus on network dynamics and network reliability (network management, mobile networks, pervasive networks)
  - Main drivers: Telecoms, Consumer Electronics
- In any case, the distinction between the two is sometimes "fuzzy"

The IBM Autonomic Computing Initiative

- Manifesto Launched in 2005
  - Motivated by the need to reduce the costs related to the configuration, optimization, healing, protection, of large ICT systems → moving humans out of the loop
  - Clearly, all large-scale and complex software systems shares the same goal (e.g., large-scale mission-critical systems)
- Quoting from the IBM manifesto:

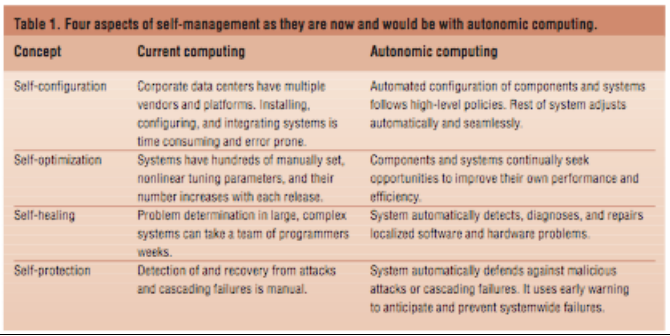
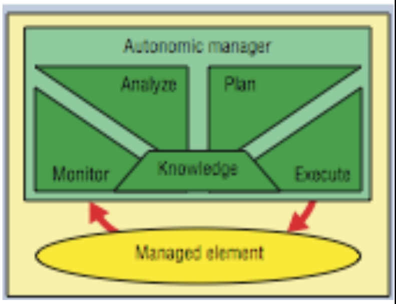


Table 1. Four aspects of self-management as they are now and would be with autonomic computing.

Concept	Current computing	Autonomic computing
Self-configuration	Corporate data centers have multiple vendors and platforms. Installing, configuring, and integrating systems is time consuming and error prone.	Automated configuration of components and systems follows high-level policies. Rest of system adjusts automatically and seamlessly.
Self-optimization	Systems have hundreds of manually set, nonlinear tuning parameters, and their number increases with each release.	Components and systems continually seek opportunities to improve their own performance and efficiency.
Self-healing	Problem determination in large, complex systems can take a team of programmers weeks.	System automatically detects, diagnoses, and repairs localized software and hardware problems.
Self-protection	Detection of and recovery from attacks and cascading failures is manual.	System automatically defends against malicious attacks or cascading failures. It uses early warning to anticipate and prevent systemwide failures.

The MAPE-K Model

- The key component suggested by IBM to achieve autonomicity is the so called "Mape-K" one
  - An element of a system is coupled with an "autonomic manager"
  - Devoted to Monitor, Analyse, Plan, Execute, based on Knowledge
  - Such that the managed component is made "autonomic"
- Directly inspired by goal-oriented agent architecture,
  - but with a more explicit "close control loop"

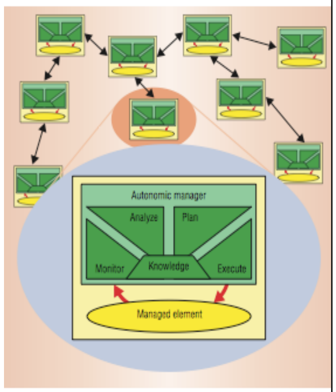


Elements of the MAPE-K Model

- Monitor
  - Gather information about the current behaviour of the component (e.g., response time, resources exploited, number of requests, etc.)
- Analyse
  - Try to understand what is happening (e.g., is fine? Are there performance problems? Are there security problems? Are there faults?)
  - Which of course requires "Knowledge", the capability of understanding data
- Plan
  - Decide corrective actions in the case of problems (e.g., gather more resources, adopt an alternate class, increase priority of execution, re-boot, etc.)
- Execute such actions on the managed element

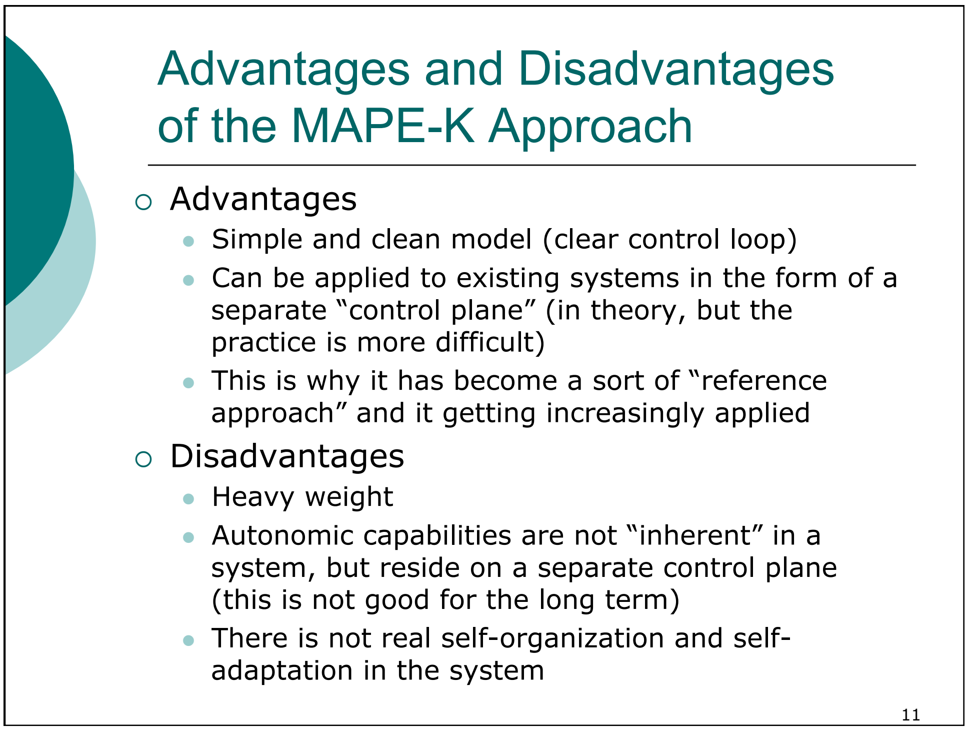
Distributed MAPE-K System

- In case the system is made up of multiple (possibly distributed) elements
  - And this is the case for many data centres, service centres, service systems
- One can think at having the different autonomic managers cooperate with each other
  - Recognition of problems involving more than one entity
  - Distributed agreement on remedial actions



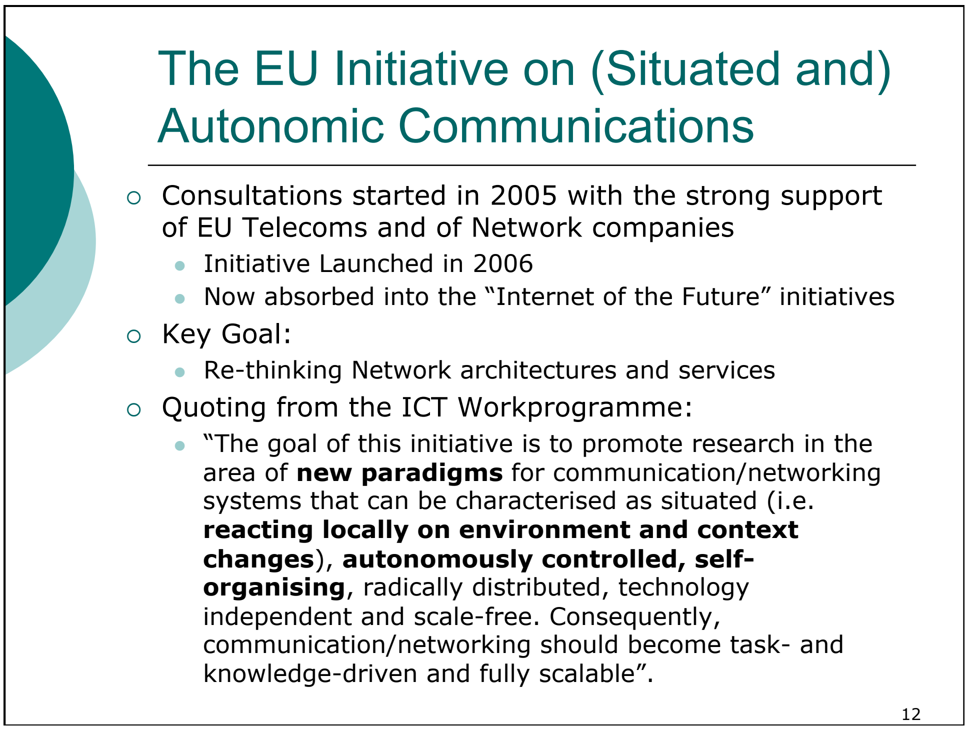
Tools for the MAPE-K recipe

- Many ideas proposed so far for…
- Autonomic managers
  - Traditional monitoring tools and probabilistic tools,…
- Analysis
  - Classifiers, bayesian reasoning, neural networks,…
- Planning
  - Logic-based ad BDI models, genetic algorithms,…
- Knowledge
  - Domain-specific ontologies
- Cooperation among autonomic managers
  - MoM and ontologies (similar to ACLs, by the way)
  - Agent-inspired negotiation mechanisms



Advantages and Disadvantages of the MAPE-K Approach

- Advantages
  - Simple and clean model (clear control loop)
  - Can be applied to existing systems in the form of a separate "control plane" (in theory, but the practice is more difficult)
  - This is why it has become a sort of "reference approach" and it getting increasingly applied
- Disadvantages
  - Heavy weight
  - Autonomic capabilities are not "inherent" in a system, but reside on a separate control plane (this is not good for the long term)
  - There is not real self-organization and self-adaptation in the system



The EU Initiative on (Situated and) Autonomic Communications

- Consultations started in 2005 with the strong support of EU Telecoms and of Network companies
  - Initiative Launched in 2006
  - Now absorbed into the "Internet of the Future" initiatives
- Key Goal:
  - Re-thinking Network architectures and services
- Quoting from the ICT Workprogramme:
  - "The goal of this initiative is to promote research in the area of **new paradigms** for communication/networking systems that can be characterised as situated (i.e. **reacting locally on environment and context changes**), **autonomously controlled, self-organising**, radically distributed, technology independent and scale-free. Consequently, communication/networking should become task- and knowledge-driven and fully scalable".

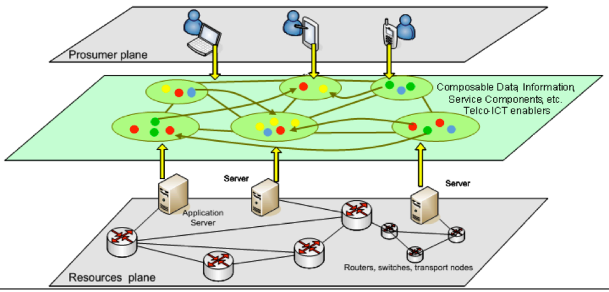
Motivations for Situated and Autonomic Communications

- Increased dynamics and scale of network scenarios
  - Mobile nodes, sensors, users,…, billions of nodes involved
- Convergence of Telecom and Internet scenarios
  - Need of a unifying approach
- Self-* features in network and network/Telecom services
  - To increase reliability and reduce management costs
- Need for decentralization
  - Not only big service/data centers and no vertical integration
- Need of services personalization
  - Accounting for users’ locations, network status, etc.
- All of the above aimed at
  - Provide better and more flexible and diverse services to users
  - At reduced costs and thus with increased revenues
- Clearly, all network systems would share this…

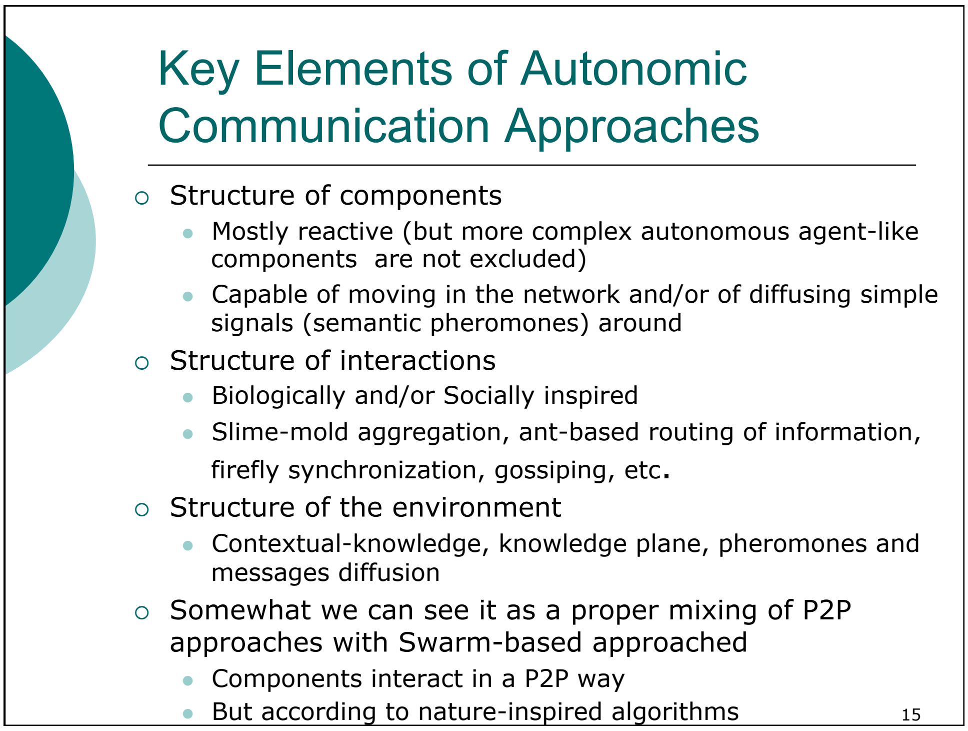
13

The Overall Approach to Autonomic Communication

- Mostly layer-less network architecture
  - All components (devices, users, producers, network agents, etc.) part of the same open “P2P plane”
  - Interacting/coordinating/linking aggregating with each other so as to dynamically self-organize and self-adapt services and functionalities
  - In a fully decentralized and unmanaged way


Prosumer plane
Composable Data, Information, Service Components, etc. Telco-ICT enablers
Server
Server
Application Server
Routers, switches, transport nodes
Resources plane

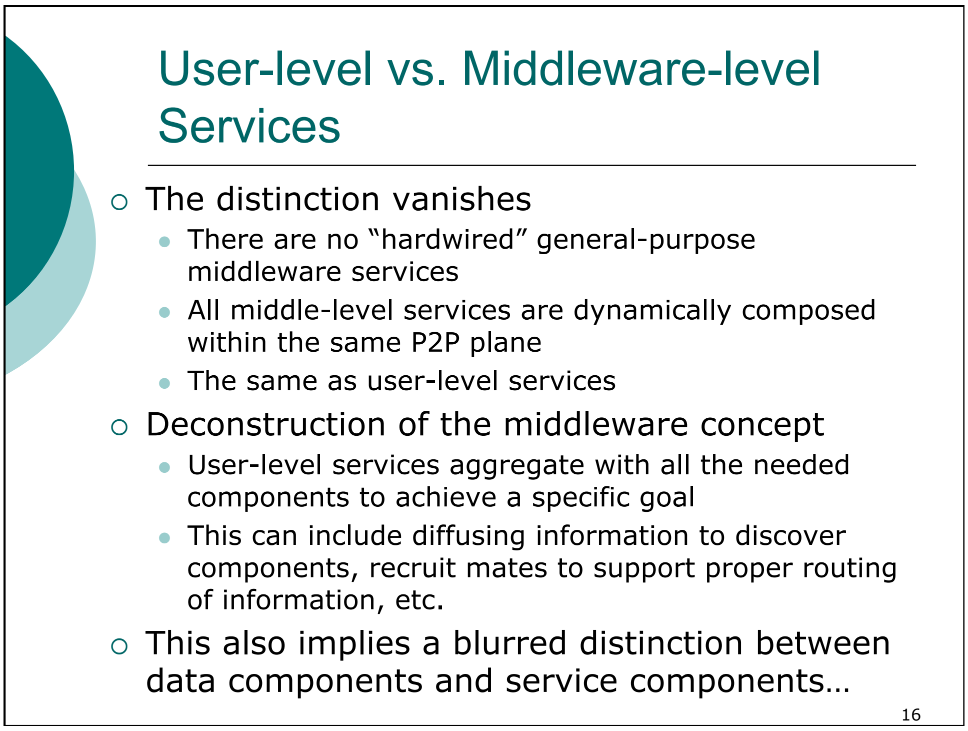
14



Key Elements of Autonomic Communication Approaches

- Structure of components
  - Mostly reactive (but more complex autonomous agent-like components are not excluded)
  - Capable of moving in the network and/or of diffusing simple signals (semantic pheromones) around
- Structure of interactions
  - Biologically and/or Socially inspired
  - Slime-mold aggregation, ant-based routing of information, firefly synchronization, gossiping, etc.
- Structure of the environment
  - Contextual-knowledge, knowledge plane, pheromones and messages diffusion
- Somewhat we can see it as a proper mixing of P2P approaches with Swarm-based approached
  - Components interact in a P2P way
  - But according to nature-inspired algorithms

15



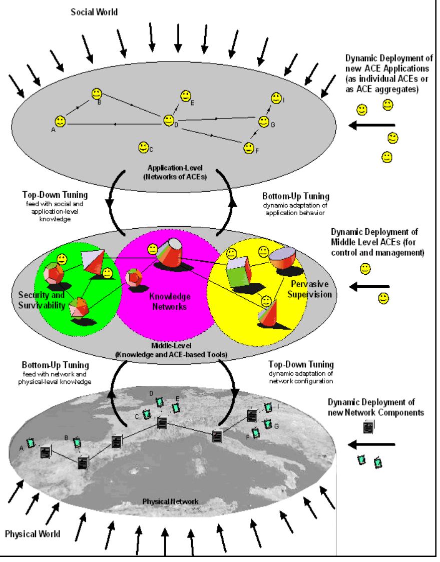
User-level vs. Middleware-level Services

- The distinction vanishes
  - There are no "hardwired" general-purpose middleware services
  - All middle-level services are dynamically composed within the same P2P plane
  - The same as user-level services
- Deconstruction of the middleware concept
  - User-level services aggregate with all the needed components to achieve a specific goal
  - This can include diffusing information to discover components, recruit mates to support proper routing of information, etc.
- This also implies a blurred distinction between data components and service components…

16

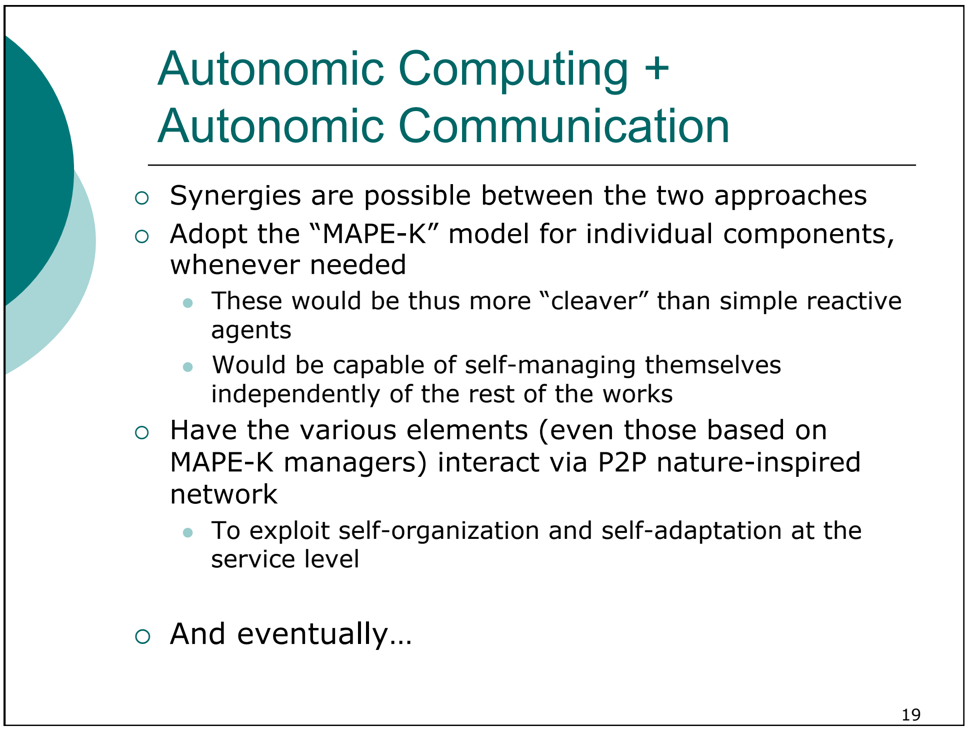
Example: The CASCADAS Project at Telecom Italia

- General objective
  - Identify models and tools for a new generation of adaptable, self-organizing, context-aware communication services
- Based on the central abstraction of ACE "autonomic communication element"
  - as the basic building block for complex service networks (both at the user-level and at the middleware level)
  - Exploiting biologically and socially-inspired self-organization and self-management phenomena


Social World
Dynamic Deployment of new ACE Applications (as individual ACEs or as ACE aggregates)
A
B
C
D
E
F
G
I
Application-Level (Networks of ACEs)
Top-Down Tuning
feed with social and application-level knowledge
Bottom-Up Tuning
dynamic adaptation of application behavior
Dynamic Deployment of Middle Level ACEs (for control and management)
Security and Survivability
Knowledge Networks
Pervasive Supervision
Middle-Level (Knowledge and ACE-based Tools)
Bottom-Up Tuning
feed with network and physical-level knowledge
Top-Down Tuning
dynamic adaptation of network configuration
Dynamic Deployment of new Network Components
Physical Network
Physical World

Advantages and Disadvantages of Autonomic Communication

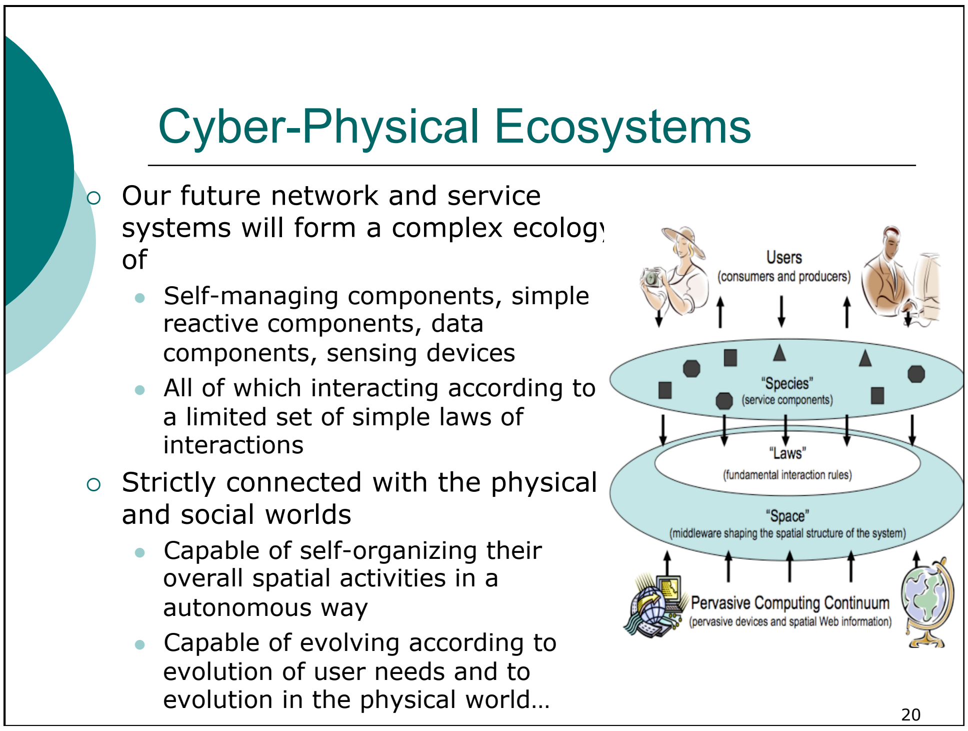
- Advantages
  - Very clean and light-weight approach, very suitable for future network scenarios
  - Potential to open brand new possibilities for the effective management of complex network systems
  - Potential to deliver new classes of personalized of services, and to open to user contributions
- Disadvantages
  - The overall vision is too much long-terms
  - Requires investments by Telecoms and Network providers
- Therefore
  - We expect the vision to be absorbed slowly, in the forms of specific nature-inspired solutions to specific problems
  - Data management, data diffusion, and data fusion problems first (easier to be released in a reliable way, and building on the lessons of existing P2P protocols)



Autonomic Computing + Autonomic Communication

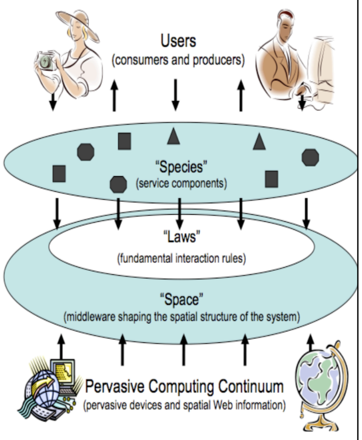
- Synergies are possible between the two approaches
- Adopt the "MAPE-K" model for individual components, whenever needed
  - These would be thus more "cleaver" than simple reactive agents
  - Would be capable of self-managing themselves independently of the rest of the works
- Have the various elements (even those based on MAPE-K managers) interact via P2P nature-inspired network
  - To exploit self-organization and self-adaptation at the service level
- And eventually…

19

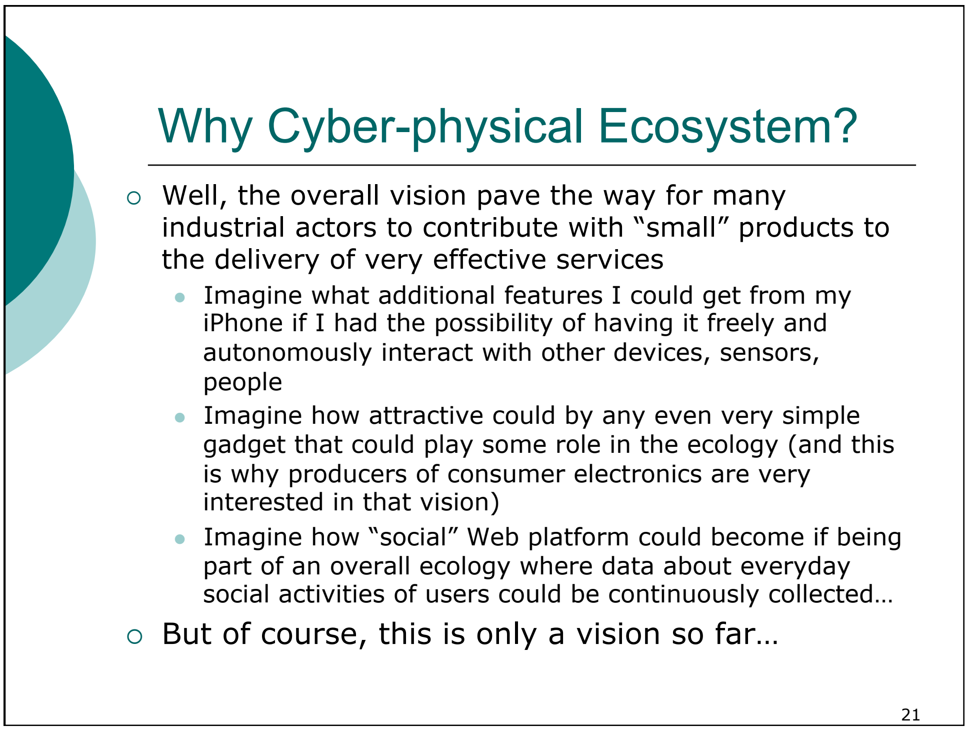


Cyber-Physical Ecosystems

- Our future network and service systems will form a complex ecolog of
  - Self-managing components, simple reactive components, data components, sensing devices
  - All of which interacting according to a limited set of simple laws of interactions
- Strictly connected with the physical and social worlds
  - Capable of self-organizing their overall spatial activities in a autonomous way
  - Capable of evolving according to evolution of user needs and to evolution in the physical world…


Users
(consumers and producers)
"Species"
(service components)
"Laws"
(fundamental interaction rules)
"Space"
(middleware shaping the spatial structure of the system)
Pervasive Computing Continuum
(pervasive devices and spatial Web information)

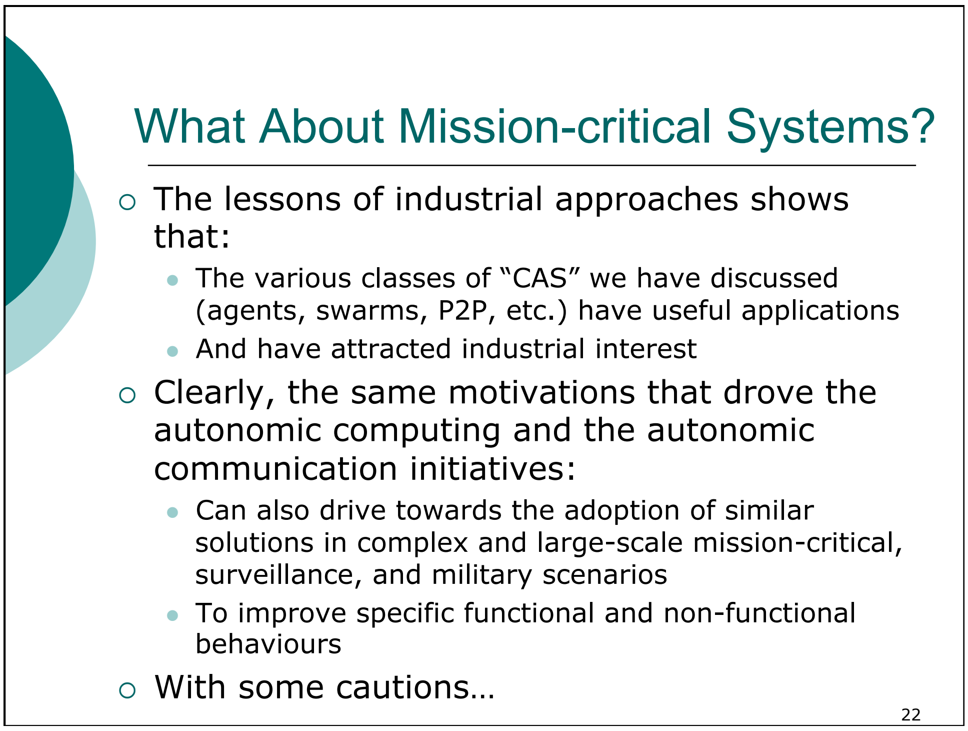
20



Why Cyber-physical Ecosystem?

- Well, the overall vision pave the way for many industrial actors to contribute with "small" products to the delivery of very effective services
  - Imagine what additional features I could get from my iPhone if I had the possibility of having it freely and autonomously interact with other devices, sensors, people
  - Imagine how attractive could by any even very simple gadget that could play some role in the ecology (and this is why producers of consumer electronics are very interested in that vision)
  - Imagine how "social" Web platform could become if being part of an overall ecology where data about everyday social activities of users could be continuously collected…
- But of course, this is only a vision so far…

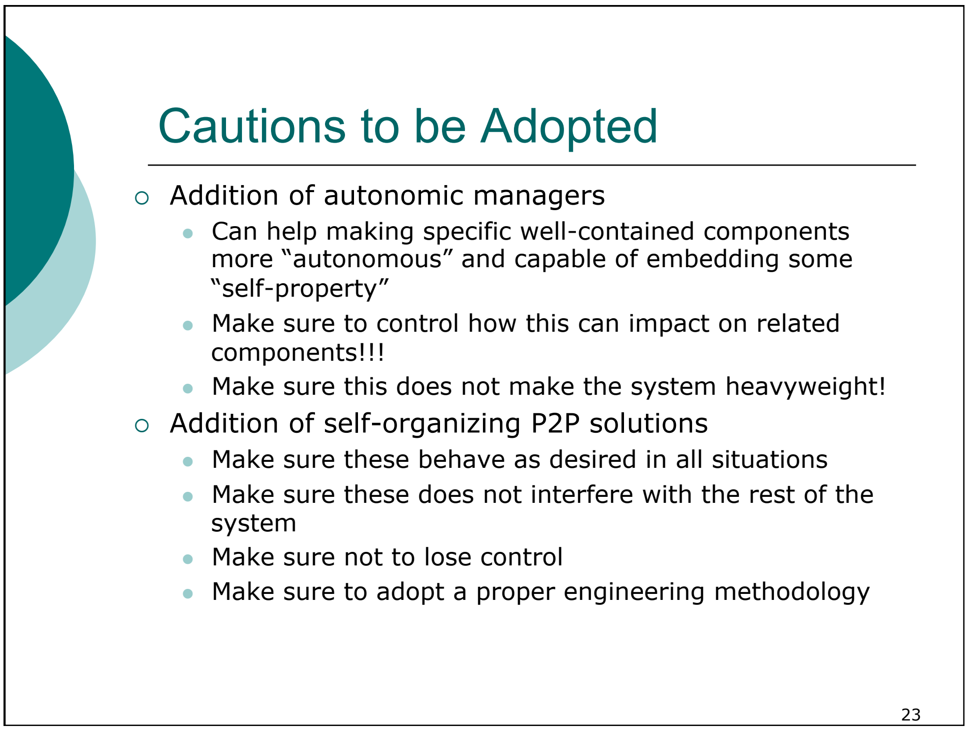
21



What About Mission-critical Systems?

- The lessons of industrial approaches shows that:
  - The various classes of "CAS" we have discussed (agents, swarms, P2P, etc.) have useful applications
  - And have attracted industrial interest
- Clearly, the same motivations that drove the autonomic computing and the autonomic communication initiatives:
  - Can also drive towards the adoption of similar solutions in complex and large-scale mission-critical, surveillance, and military scenarios
  - To improve specific functional and non-functional behaviours
- With some cautions…

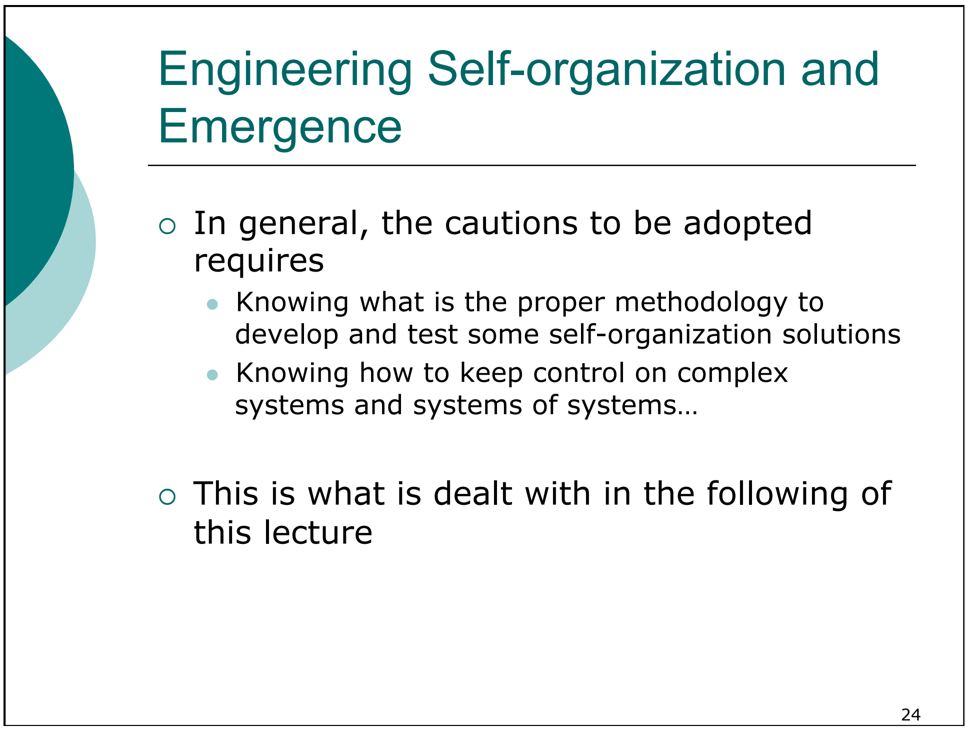
22



Cautions to be Adopted

- Addition of autonomic managers
  - Can help making specific well-contained components more "autonomous" and capable of embedding some "self-property"
  - Make sure to control how this can impact on related components!!!
  - Make sure this does not make the system heavyweight!
- Addition of self-organizing P2P solutions
  - Make sure these behave as desired in all situations
  - Make sure these does not interfere with the rest of the system
  - Make sure not to lose control
  - Make sure to adopt a proper engineering methodology

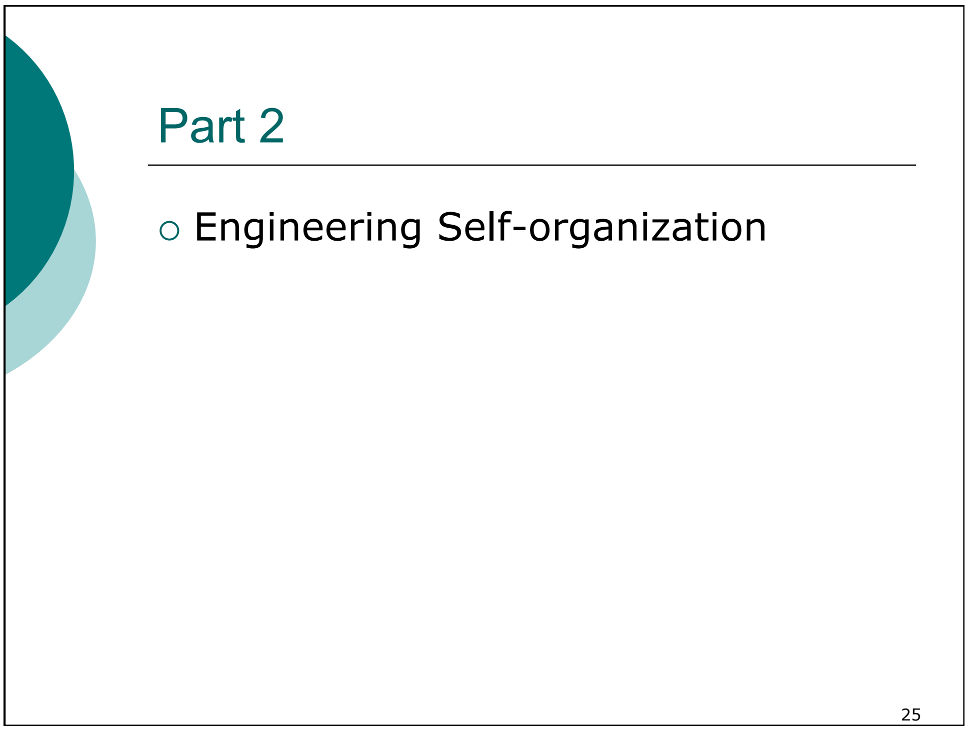
23



Engineering Self-organization and Emergence

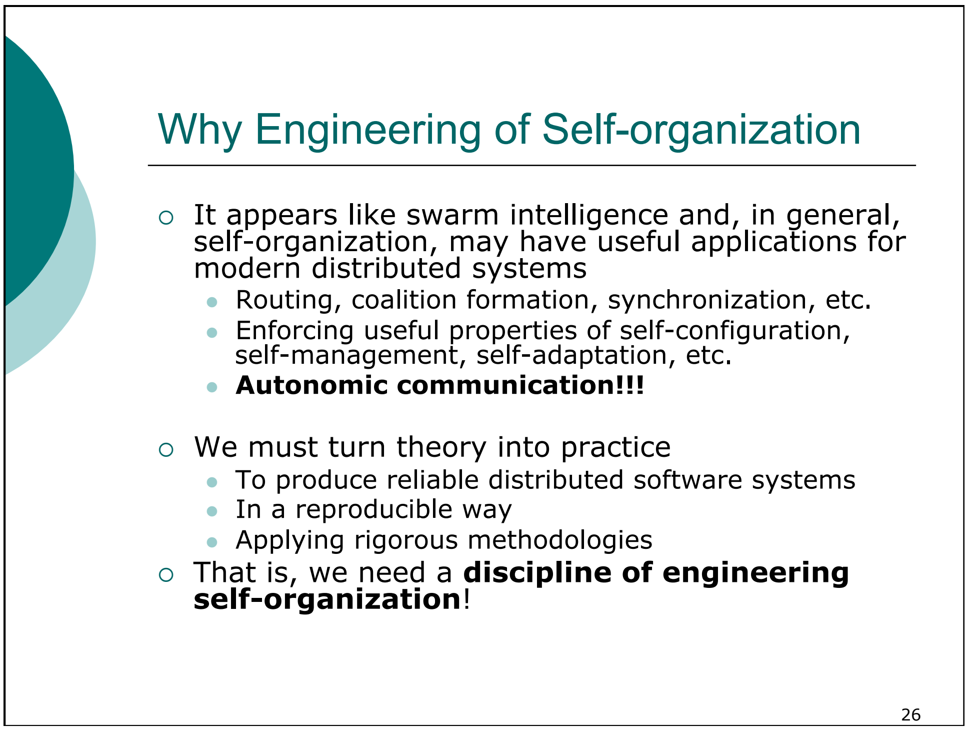
- In general, the cautions to be adopted requires
  - Knowing what is the proper methodology to develop and test some self-organization solutions
  - Knowing how to keep control on complex systems and systems of systems…
- This is what is dealt with in the following of this lecture

24



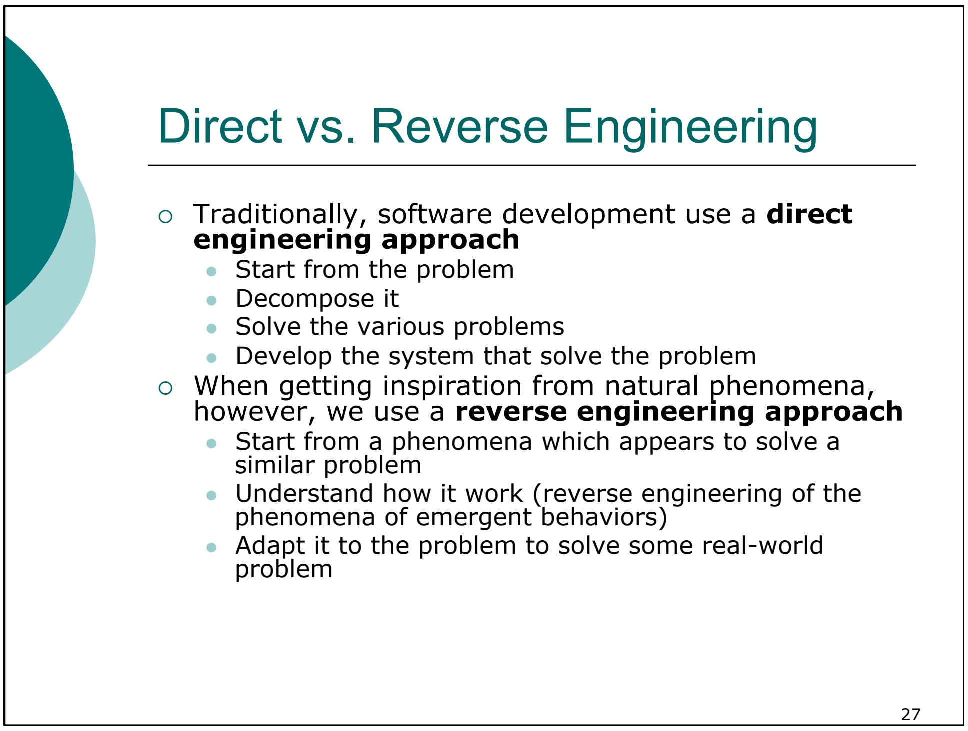
Part 2

- Engineering Self-organization



Why Engineering of Self-organization

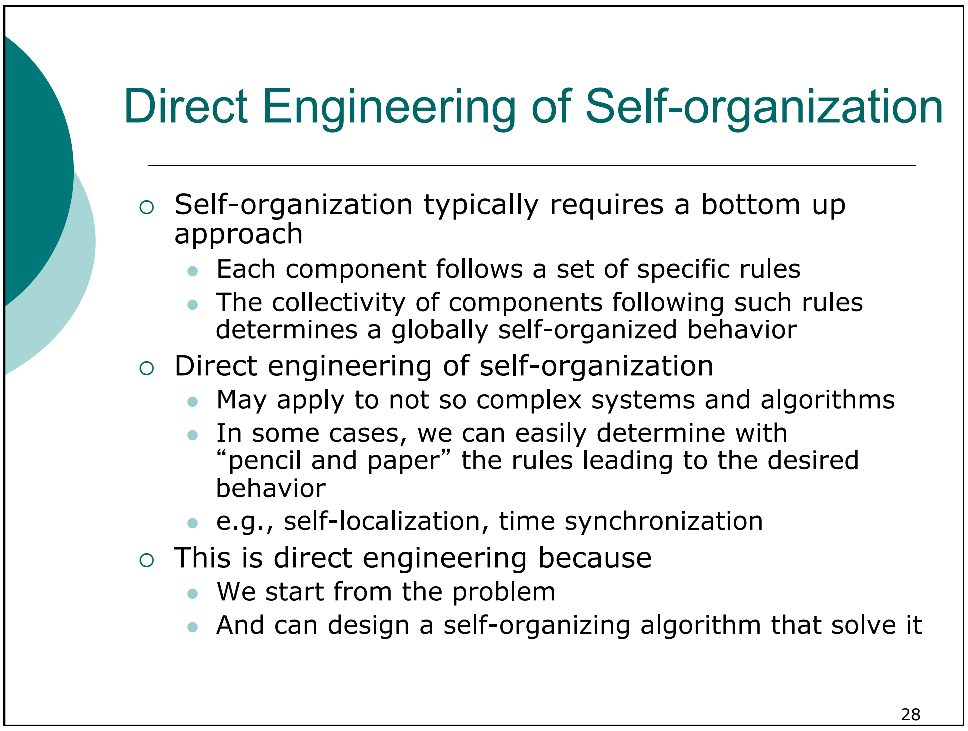
- It appears like swarm intelligence and, in general, self-organization, may have useful applications for modern distributed systems
  - Routing, coalition formation, synchronization, etc.
  - Enforcing useful properties of self-configuration, self-management, self-adaptation, etc.
  - **Autonomic communication!!!**
- We must turn theory into practice
  - To produce reliable distributed software systems
  - In a reproducible way
  - Applying rigorous methodologies
- That is, we need a **discipline of engineering self-organization**!



Direct vs. Reverse Engineering

- Traditionally, software development use a **direct engineering approach**
  - Start from the problem
  - Decompose it
  - Solve the various problems
  - Develop the system that solve the problem
- When getting inspiration from natural phenomena, however, we use a **reverse engineering approach**
  - Start from a phenomena which appears to solve a similar problem
  - Understand how it work (reverse engineering of the phenomena of emergent behaviors)
  - Adapt it to the problem to solve some real-world problem

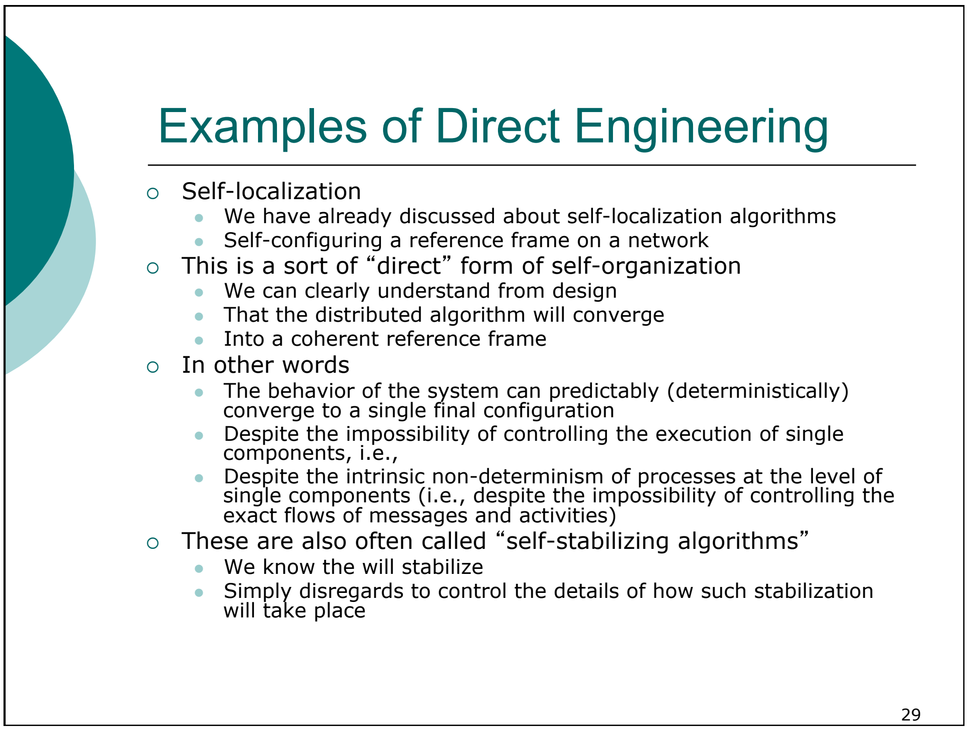
27



Direct Engineering of Self-organization

- Self-organization typically requires a bottom up approach
  - Each component follows a set of specific rules
  - The collectivity of components following such rules determines a globally self-organized behavior
- Direct engineering of self-organization
  - May apply to not so complex systems and algorithms
  - In some cases, we can easily determine with "pencil and paper" the rules leading to the desired behavior
  - e.g., self-localization, time synchronization
- This is direct engineering because
  - We start from the problem
  - And can design a self-organizing algorithm that solve it

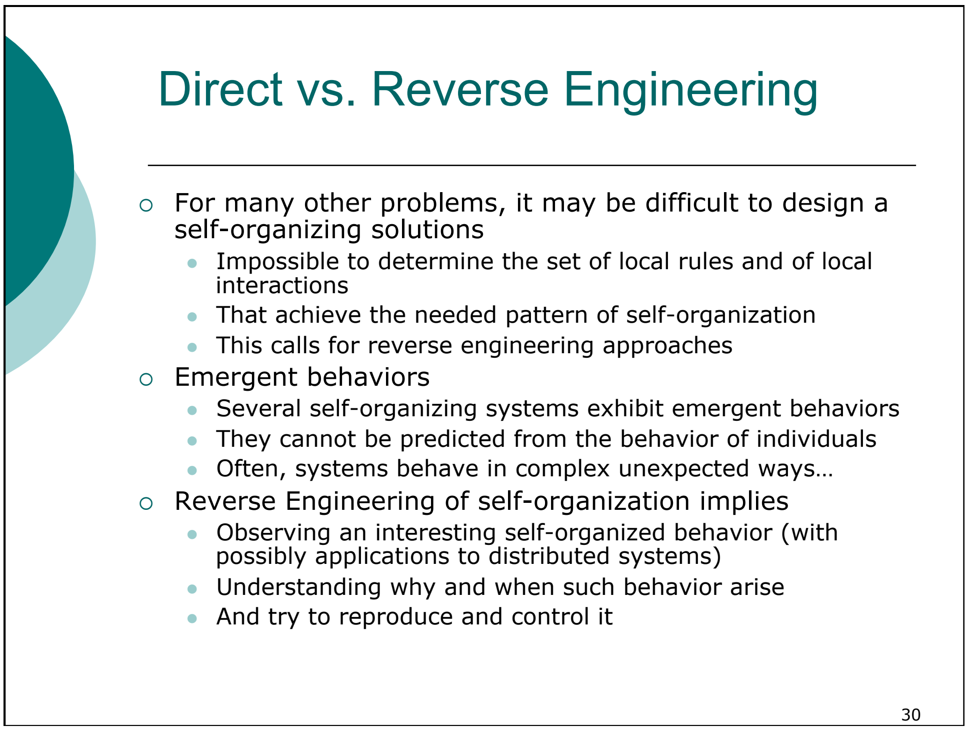
28



Examples of Direct Engineering

- Self-localization
  - We have already discussed about self-localization algorithms
  - Self-configuring a reference frame on a network
- This is a sort of “direct” form of self-organization
  - We can clearly understand from design
  - That the distributed algorithm will converge
  - Into a coherent reference frame
- In other words
  - The behavior of the system can predictably (deterministically) converge to a single final configuration
  - Despite the impossibility of controlling the execution of single components, i.e.,
  - Despite the intrinsic non-determinism of processes at the level of single components (i.e., despite the impossibility of controlling the exact flows of messages and activities)
- These are also often called “self-stabilizing algorithms”
  - We know the will stabilize
  - Simply disregards to control the details of how such stabilization will take place

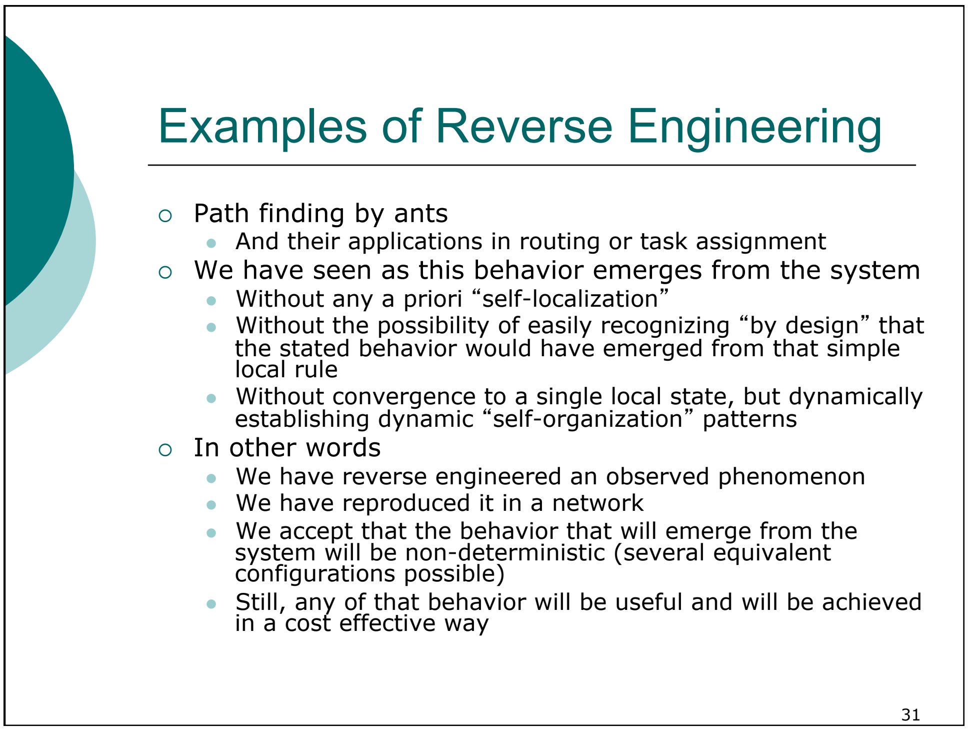
29



Direct vs. Reverse Engineering

- For many other problems, it may be difficult to design a self-organizing solutions
  - Impossible to determine the set of local rules and of local interactions
  - That achieve the needed pattern of self-organization
  - This calls for reverse engineering approaches
- Emergent behaviors
  - Several self-organizing systems exhibit emergent behaviors
  - They cannot be predicted from the behavior of individuals
  - Often, systems behave in complex unexpected ways…
- Reverse Engineering of self-organization implies
  - Observing an interesting self-organized behavior (with possibly applications to distributed systems)
  - Understanding why and when such behavior arise
  - And try to reproduce and control it

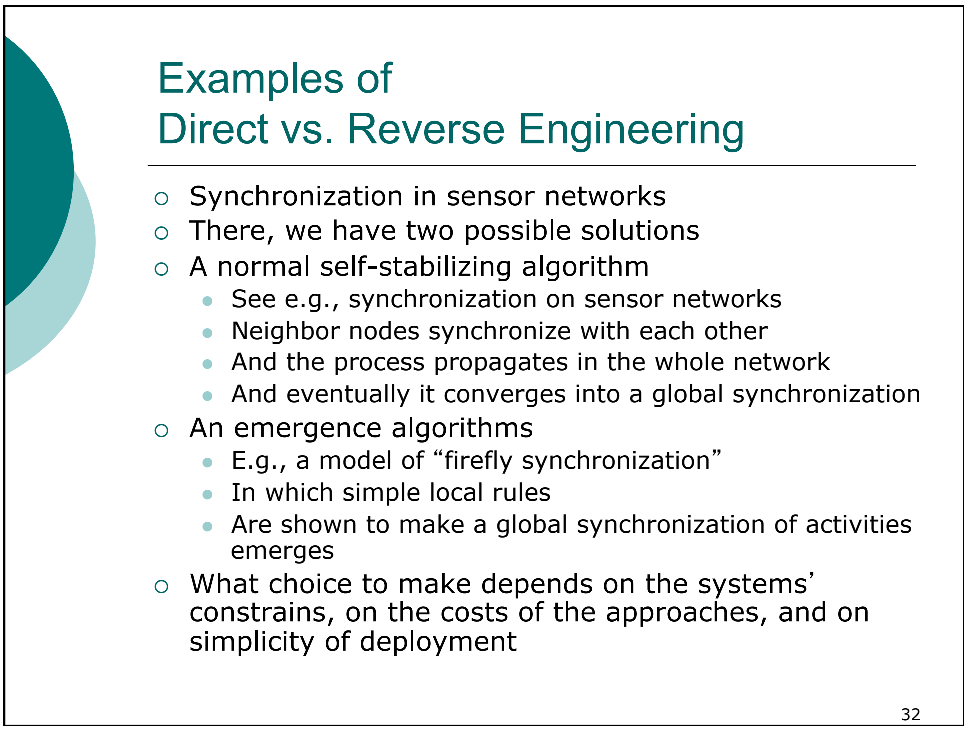
30



Examples of Reverse Engineering

- Path finding by ants
  - And their applications in routing or task assignment
- We have seen as this behavior emerges from the system
  - Without any a priori “self-localization”
  - Without the possibility of easily recognizing “by design” that the stated behavior would have emerged from that simple local rule
  - Without convergence to a single local state, but dynamically establishing dynamic “self-organization” patterns
- In other words
  - We have reverse engineered an observed phenomenon
  - We have reproduced it in a network
  - We accept that the behavior that will emerge from the system will be non-deterministic (several equivalent configurations possible)
  - Still, any of that behavior will be useful and will be achieved in a cost effective way

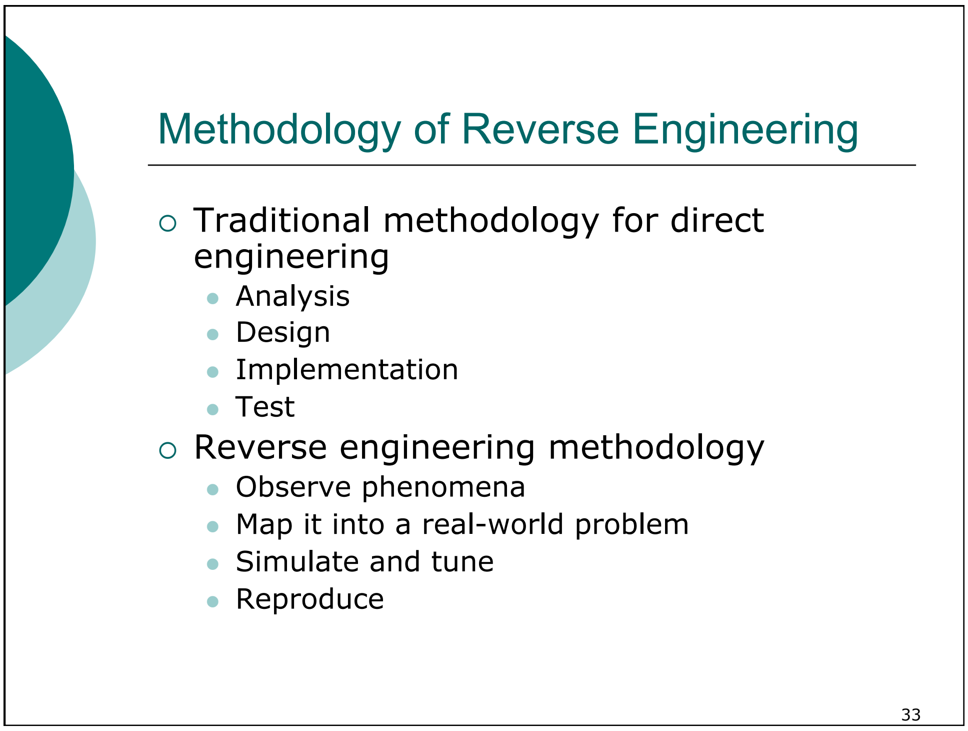
31



Examples of Direct vs. Reverse Engineering

- Synchronization in sensor networks
- There, we have two possible solutions
- A normal self-stabilizing algorithm
  - See e.g., synchronization on sensor networks
  - Neighbor nodes synchronize with each other
  - And the process propagates in the whole network
  - And eventually it converges into a global synchronization
- An emergence algorithms
  - E.g., a model of “firefly synchronization”
  - In which simple local rules
  - Are shown to make a global synchronization of activities emerges
- What choice to make depends on the systems’ constrains, on the costs of the approaches, and on simplicity of deployment

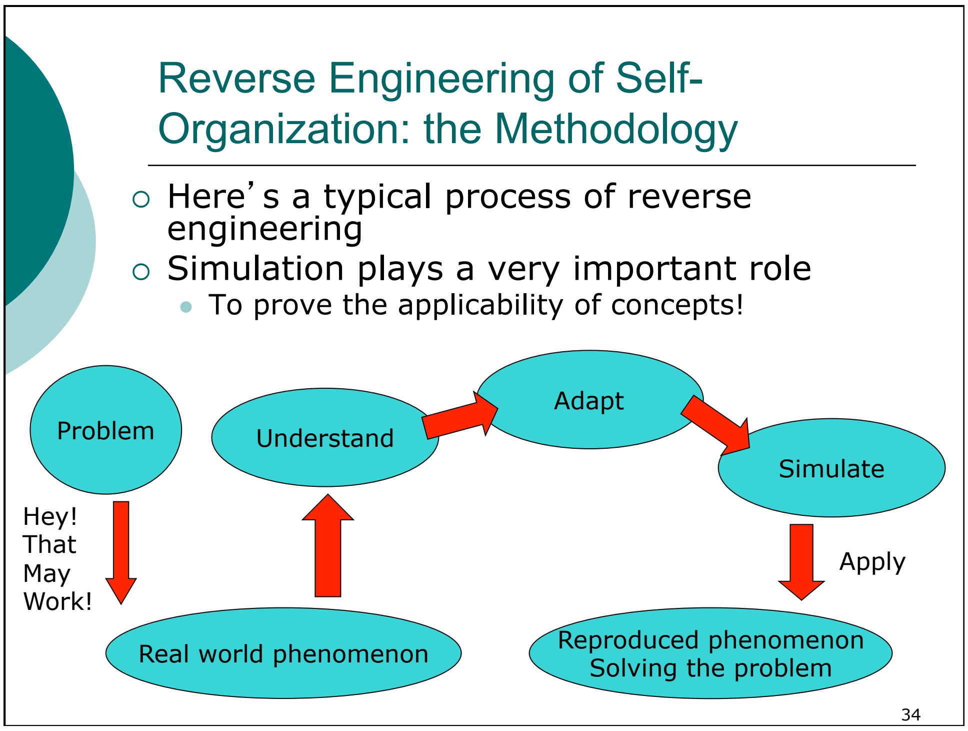
32



Methodology of Reverse Engineering

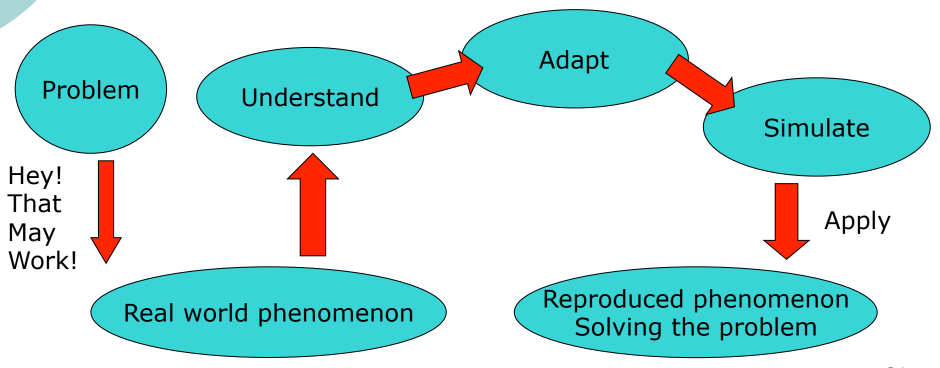
- Traditional methodology for direct engineering
  - Analysis
  - Design
  - Implementation
  - Test
- Reverse engineering methodology
  - Observe phenomena
  - Map it into a real-world problem
  - Simulate and tune
  - Reproduce

33

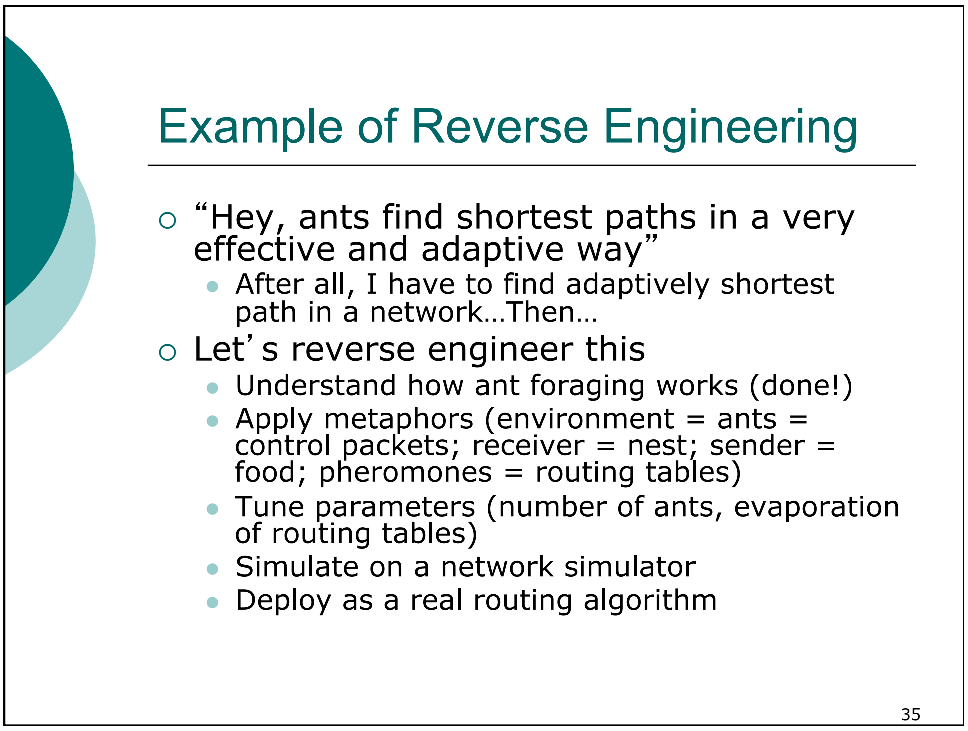


Reverse Engineering of Self-Organization: the Methodology

- Here's a typical process of reverse engineering
- Simulation plays a very important role
  - To prove the applicability of concepts!



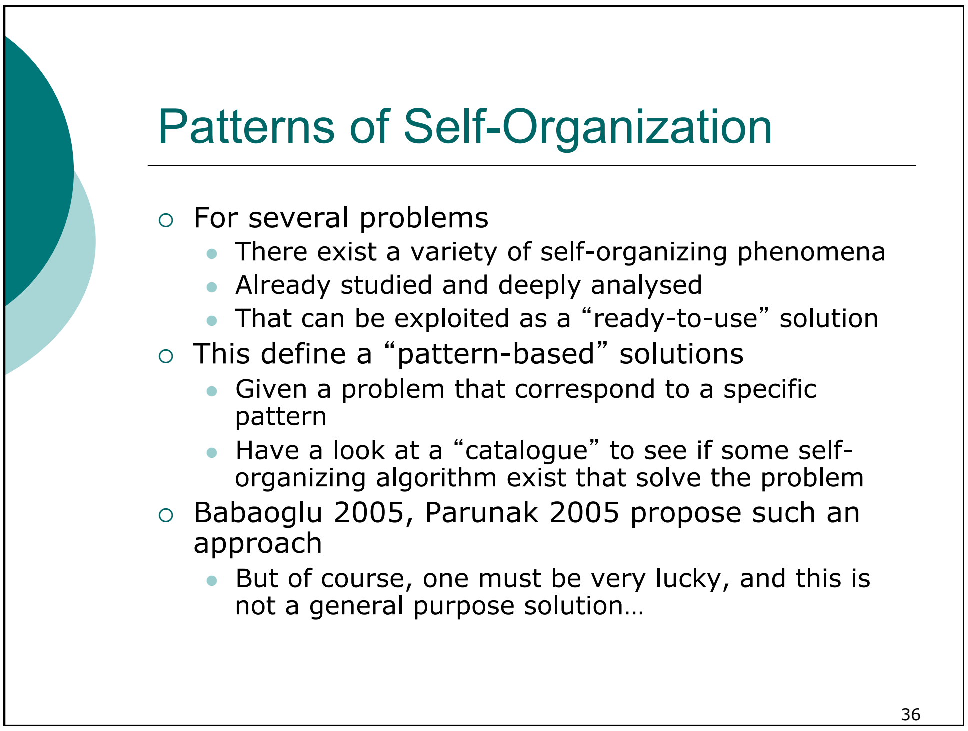
34



Example of Reverse Engineering

- "Hey, ants find shortest paths in a very effective and adaptive way"
  - After all, I have to find adaptively shortest path in a network…Then…
- Let's reverse engineer this
  - Understand how ant foraging works (done!)
  - Apply metaphors (environment = ants = control packets; receiver = nest; sender = food; pheromones = routing tables)
  - Tune parameters (number of ants, evaporation of routing tables)
  - Simulate on a network simulator
  - Deploy as a real routing algorithm

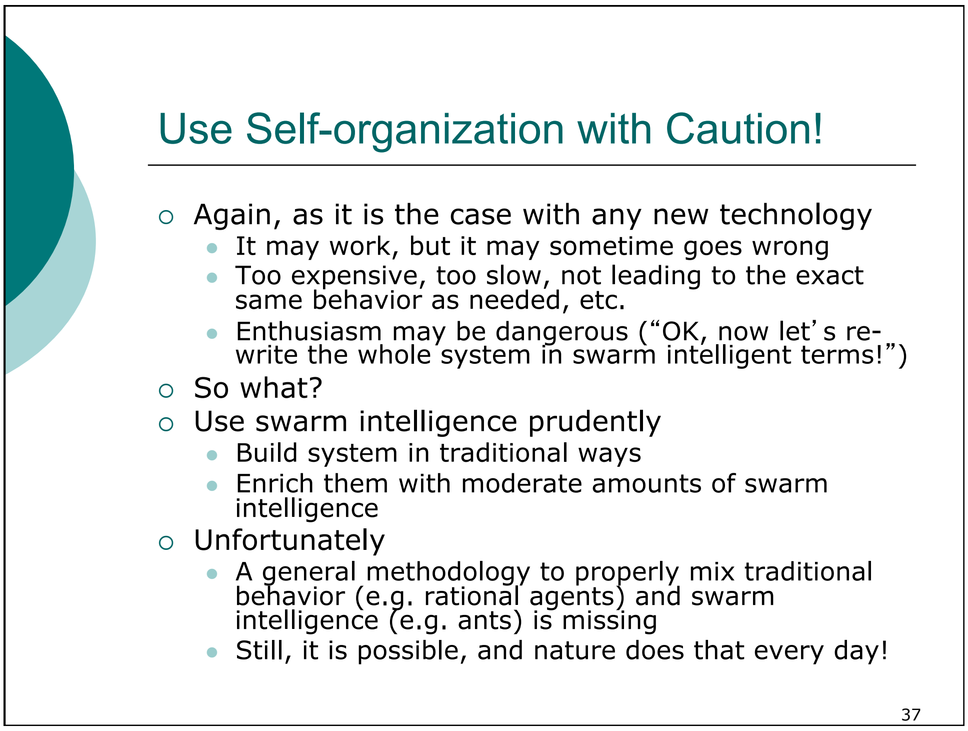
35



Patterns of Self-Organization

- For several problems
  - There exist a variety of self-organizing phenomena
  - Already studied and deeply analysed
  - That can be exploited as a "ready-to-use" solution
- This define a "pattern-based" solutions
  - Given a problem that correspond to a specific pattern
  - Have a look at a "catalogue" to see if some self-organizing algorithm exist that solve the problem
- Babaoglu 2005, Parunak 2005 propose such an approach
  - But of course, one must be very lucky, and this is not a general purpose solution…

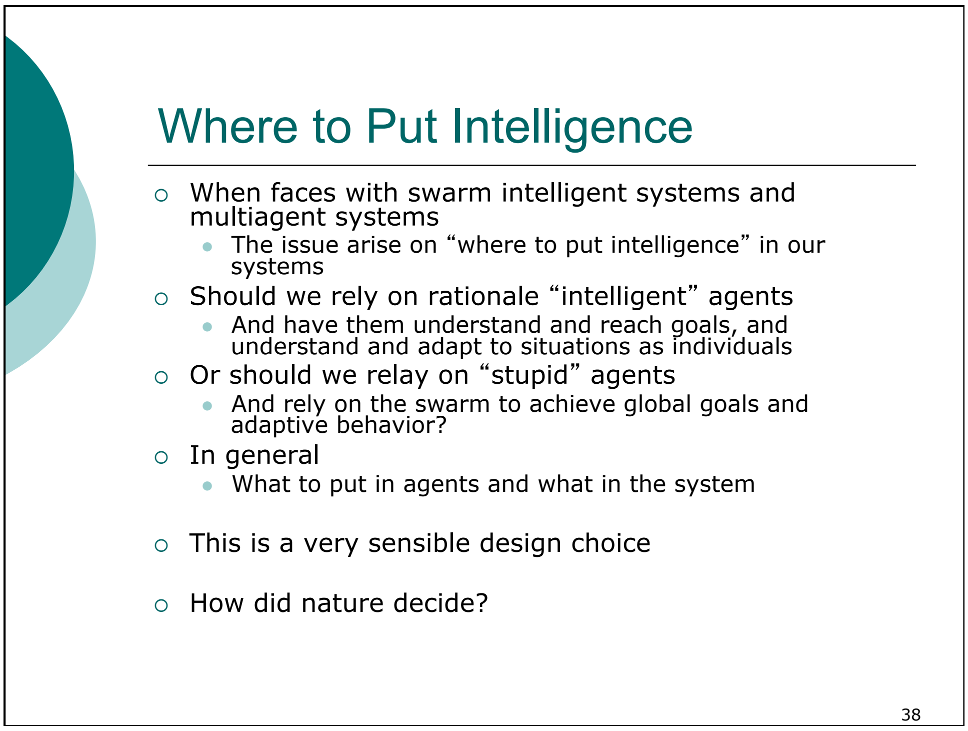
36



Use Self-organization with Caution!

- Again, as it is the case with any new technology
  - It may work, but it may sometime goes wrong
  - Too expensive, too slow, not leading to the exact same behavior as needed, etc.
  - Enthusiasm may be dangerous ("OK, now let's re-write the whole system in swarm intelligent terms!")
- So what?
- Use swarm intelligence prudently
  - Build system in traditional ways
  - Enrich them with moderate amounts of swarm intelligence
- Unfortunately
  - A general methodology to properly mix traditional behavior (e.g. rational agents) and swarm intelligence (e.g. ants) is missing
  - Still, it is possible, and nature does that every day!

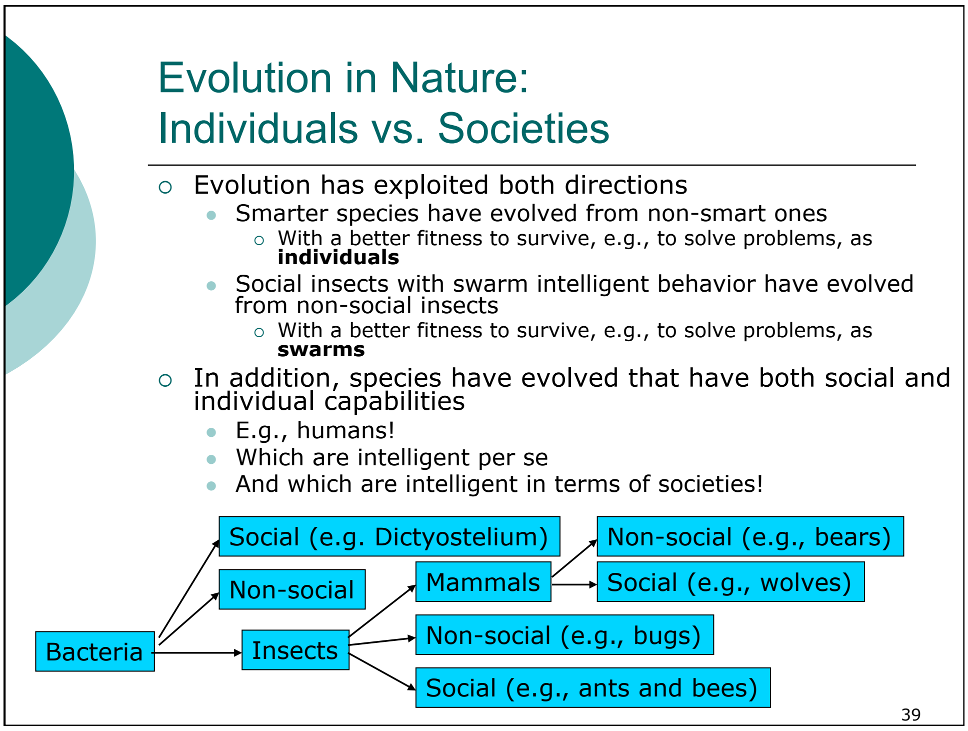
37



Where to Put Intelligence

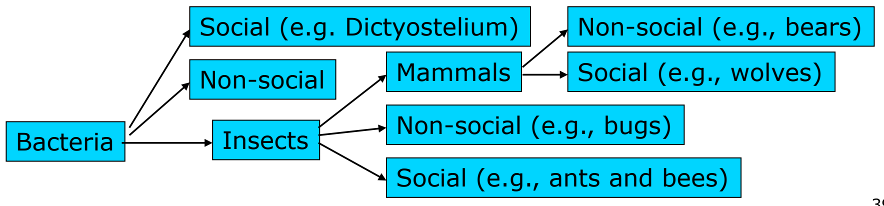
- When faces with swarm intelligent systems and multiagent systems
  - The issue arise on "where to put intelligence" in our systems
- Should we rely on rationale "intelligent" agents
  - And have them understand and reach goals, and understand and adapt to situations as individuals
- Or should we relay on "stupid" agents
  - And rely on the swarm to achieve global goals and adaptive behavior?
- In general
  - What to put in agents and what in the system
- This is a very sensible design choice
- How did nature decide?

38

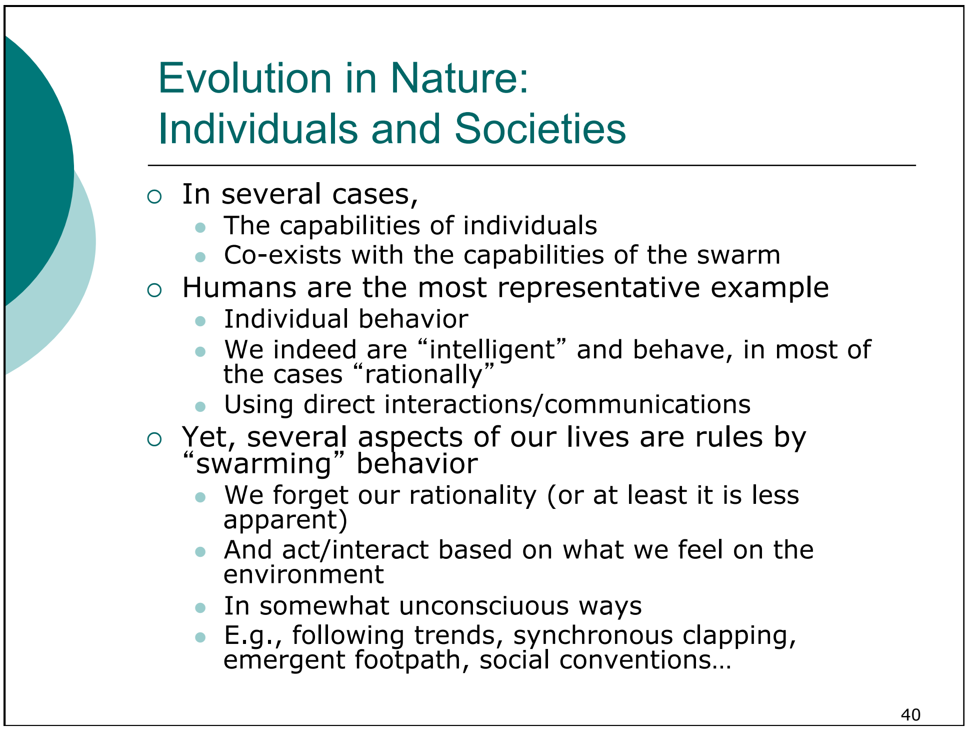


Evolution in Nature: Individuals vs. Societies

- Evolution has exploited both directions
  - Smarter species have evolved from non-smart ones
    - With a better fitness to survive, e.g., to solve problems, as **individuals**
  - Social insects with swarm intelligent behavior have evolved from non-social insects
    - With a better fitness to survive, e.g., to solve problems, as **swarms**
- In addition, species have evolved that have both social and individual capabilities
  - E.g., humans!
  - Which are intelligent per se
  - And which are intelligent in terms of societies!



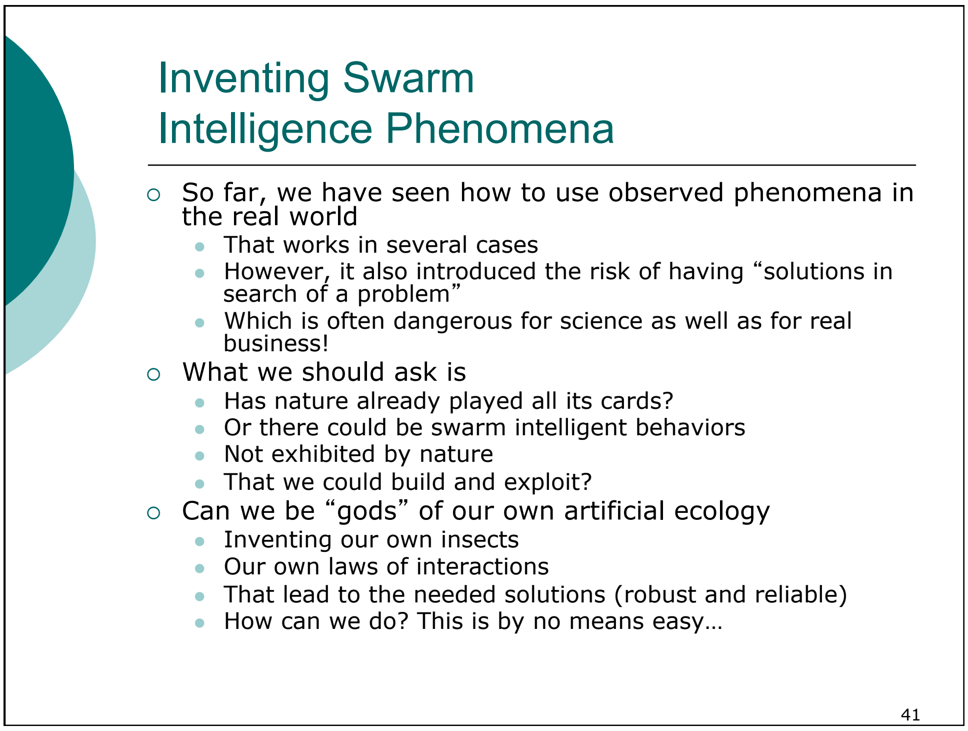
39



Evolution in Nature: Individuals and Societies

- In several cases,
  - The capabilities of individuals
  - Co-exists with the capabilities of the swarm
- Humans are the most representative example
  - Individual behavior
  - We indeed are “intelligent” and behave, in most of the cases “rationally”
  - Using direct interactions/communications
- Yet, several aspects of our lives are rules by “swarming” behavior
  - We forget our rationality (or at least it is less apparent)
  - And act/interact based on what we feel on the environment
  - In somewhat unconsciuous ways
  - E.g., following trends, synchronous clapping, emergent footpath, social conventions…

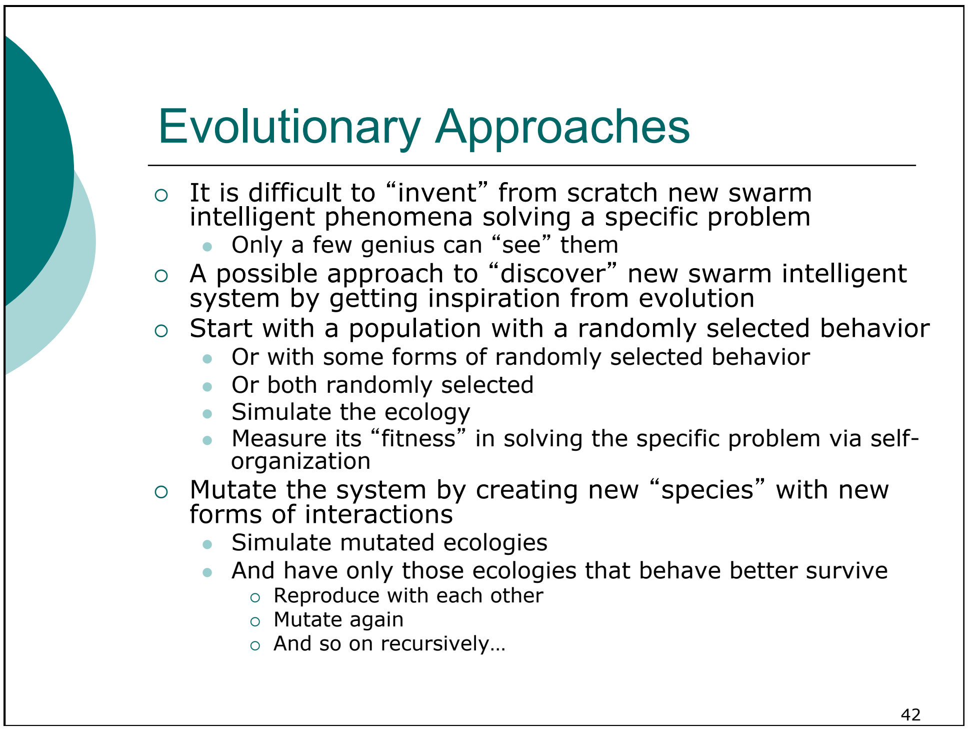
40



Inventing Swarm Intelligence Phenomena

- So far, we have seen how to use observed phenomena in the real world
  - That works in several cases
  - However, it also introduced the risk of having “solutions in search of a problem”
  - Which is often dangerous for science as well as for real business!
- What we should ask is
  - Has nature already played all its cards?
  - Or there could be swarm intelligent behaviors
  - Not exhibited by nature
  - That we could build and exploit?
- Can we be “gods” of our own artificial ecology
  - Inventing our own insects
  - Our own laws of interactions
  - That lead to the needed solutions (robust and reliable)
  - How can we do? This is by no means easy…

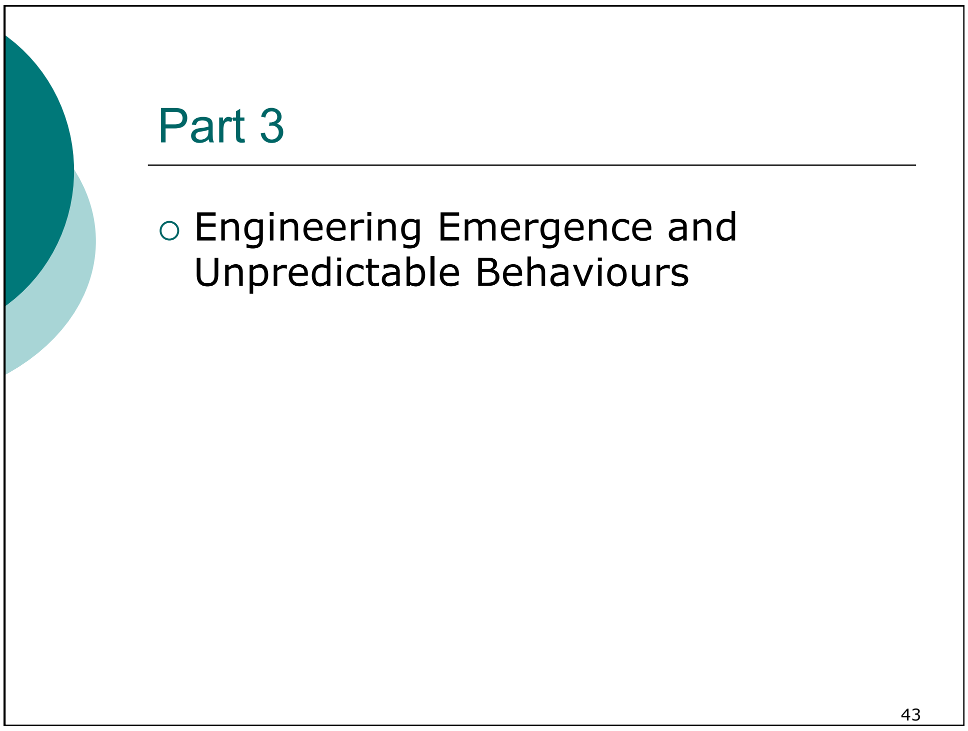
41



Evolutionary Approaches

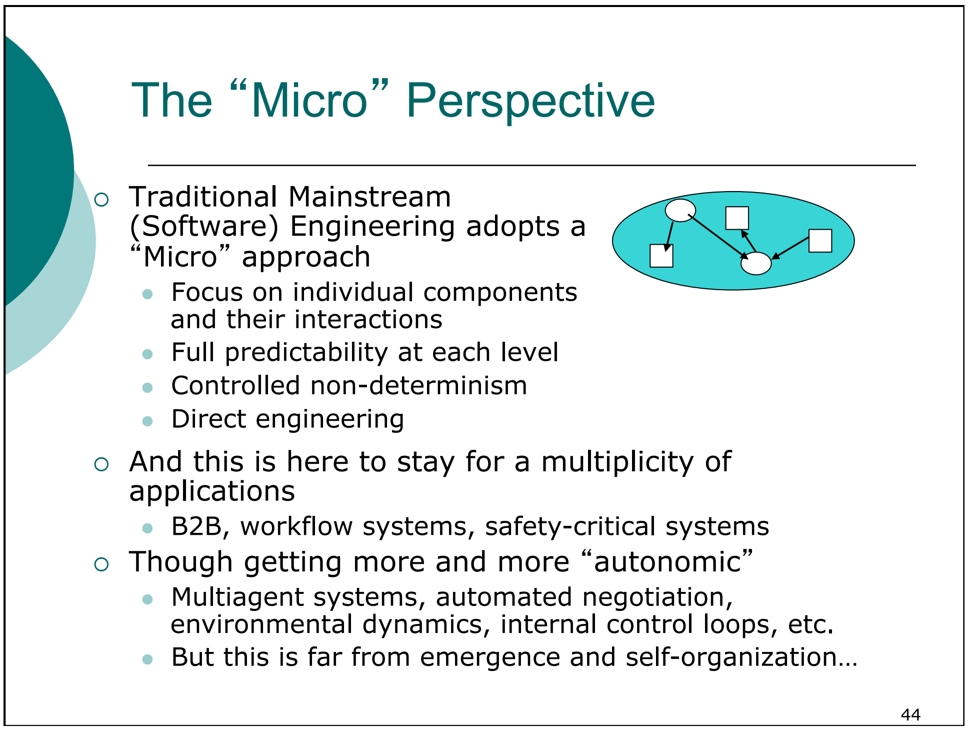
- It is difficult to “invent” from scratch new swarm intelligent phenomena solving a specific problem
  - Only a few genius can “see” them
- A possible approach to “discover” new swarm intelligent system by getting inspiration from evolution
- Start with a population with a randomly selected behavior
  - Or with some forms of randomly selected behavior
  - Or both randomly selected
  - Simulate the ecology
  - Measure its “fitness” in solving the specific problem via self-organization
- Mutate the system by creating new “species” with new forms of interactions
  - Simulate mutated ecologies
  - And have only those ecologies that behave better survive
    - Reproduce with each other
    - Mutate again
    - And so on recursively…

42



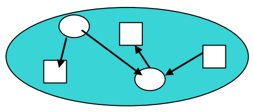
Part 3

- Engineering Emergence and Unpredictable Behaviours

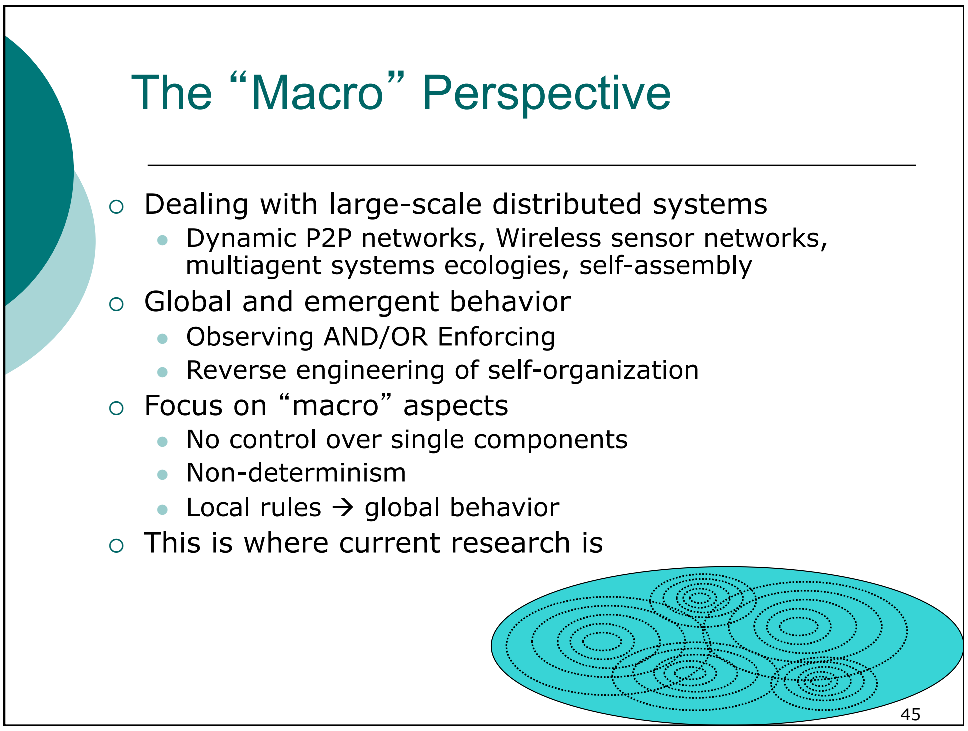


The “Micro” Perspective

- Traditional Mainstream (Software) Engineering adopts a “Micro” approach
  - Focus on individual components and their interactions
  - Full predictability at each level
  - Controlled non-determinism
  - Direct engineering

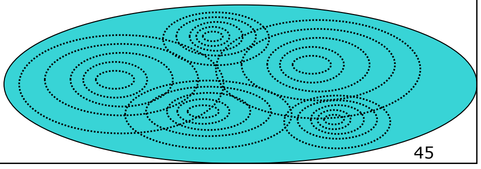


- And this is here to stay for a multiplicity of applications
  - B2B, workflow systems, safety-critical systems
- Though getting more and more “autonomic”
  - Multiagent systems, automated negotiation, environmental dynamics, internal control loops, etc.
  - But this is far from emergence and self-organization…

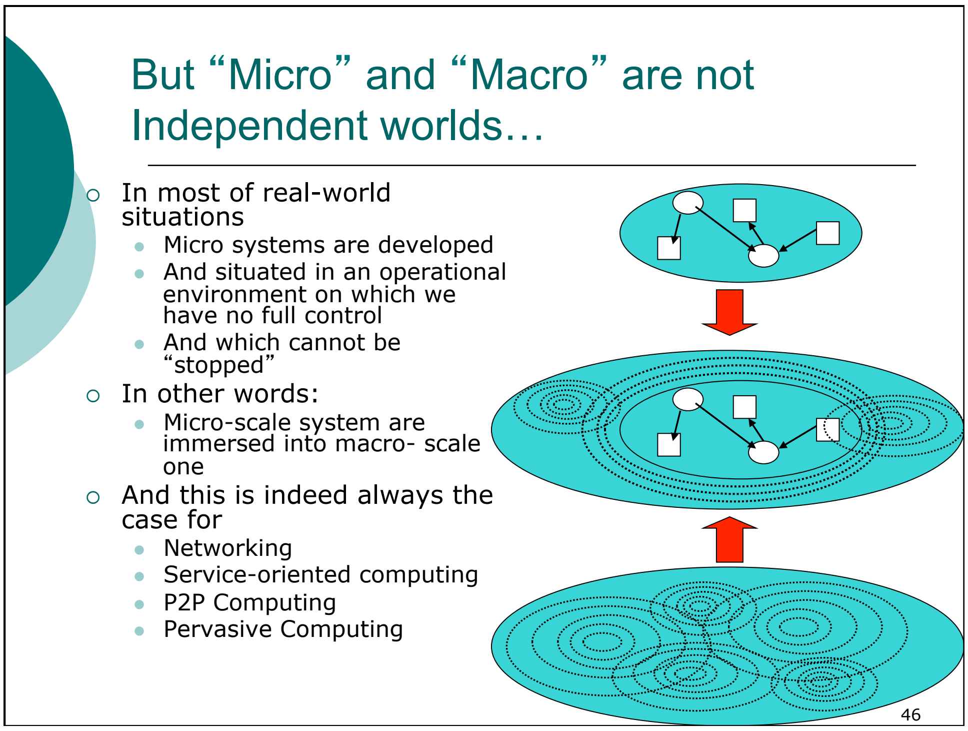


The “Macro” Perspective

- Dealing with large-scale distributed systems
  - Dynamic P2P networks, Wireless sensor networks, multiagent systems ecologies, self-assembly
- Global and emergent behavior
  - Observing AND/OR Enforcing
  - Reverse engineering of self-organization
- Focus on “macro” aspects
  - No control over single components
  - Non-determinism
  - Local rules → global behavior
- This is where current research is

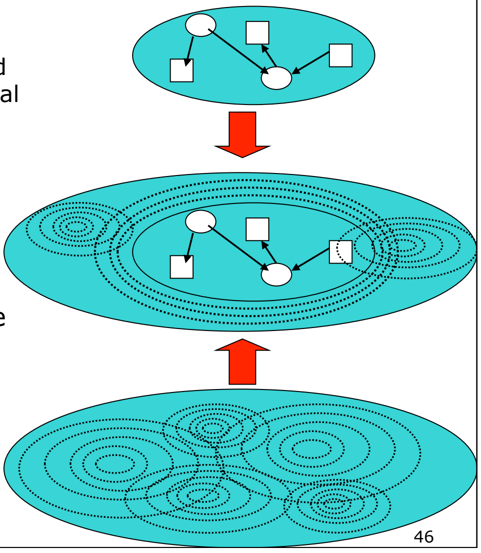


45

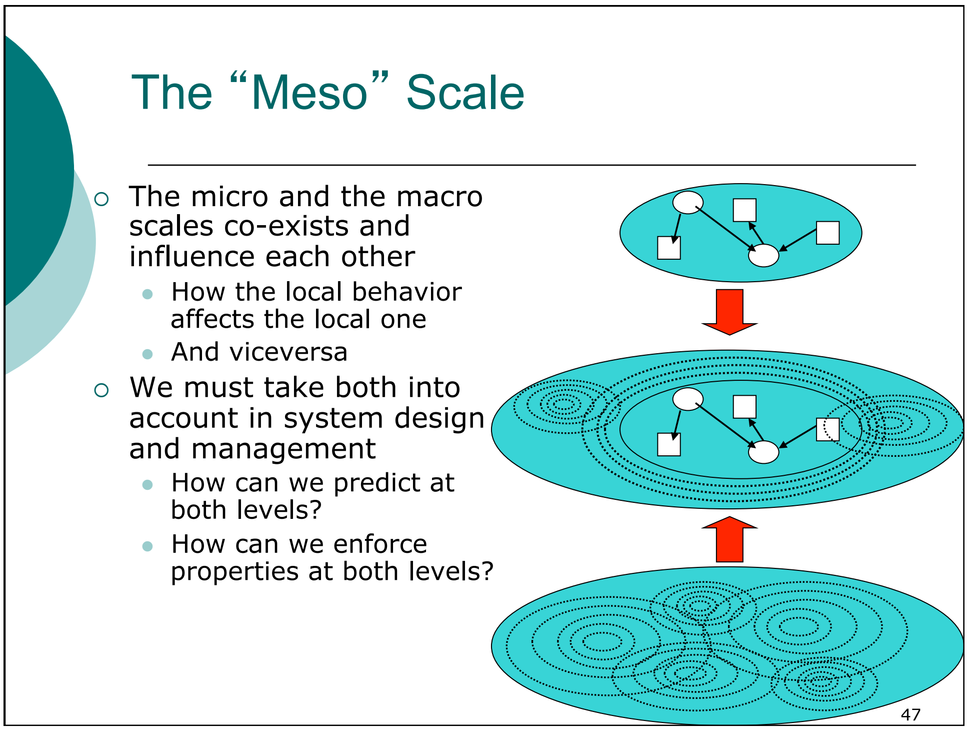


But “Micro” and “Macro” are not Independent worlds…

- In most of real-world situations
  - Micro systems are developed
  - And situated in an operational environment on which we have no full control
  - And which cannot be “stopped”
- In other words:
  - Micro-scale system are immersed into macro- scale one
- And this is indeed always the case for
  - Networking
  - Service-oriented computing
  - P2P Computing
  - Pervasive Computing

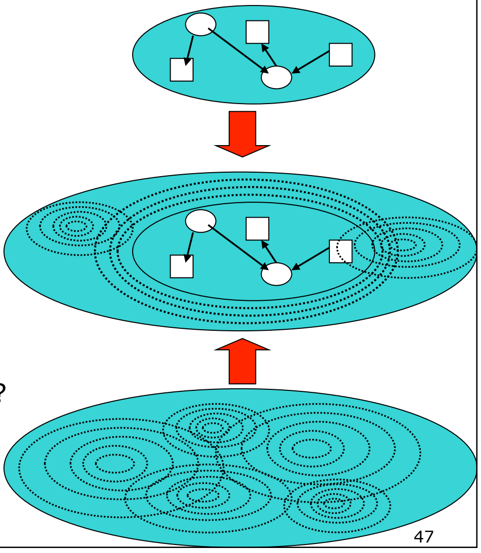


46

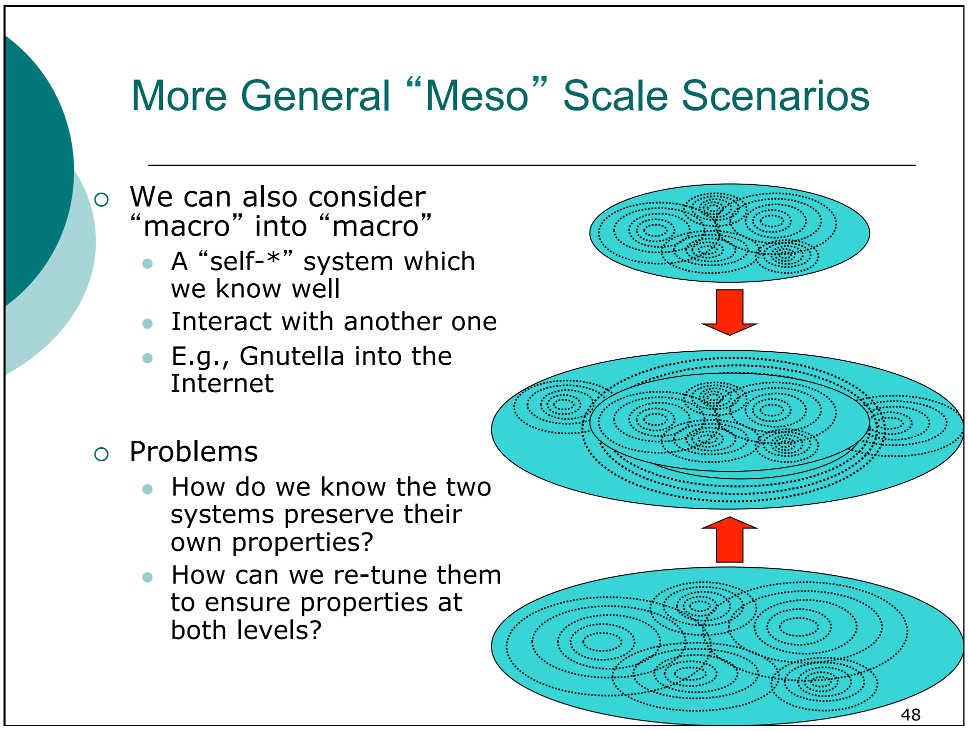


The "Meso" Scale

- The micro and the macro scales co-exists and influence each other
  - How the local behavior affects the local one
  - And viceversa
- We must take both into account in system design and management
  - How can we predict at both levels?
  - How can we enforce properties at both levels?

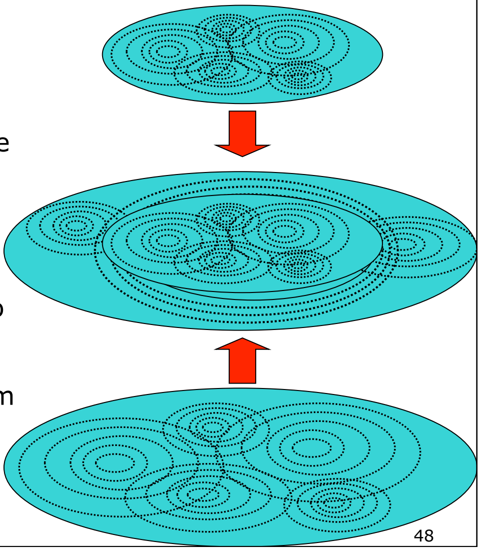


47



More General "Meso" Scale Scenarios

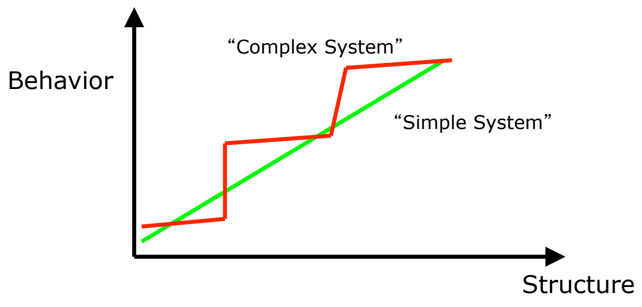
- We can also consider "macro" into "macro"
  - A "self-*" system which we know well
  - Interact with another one
  - E.g., Gnutella into the Internet
- Problems
  - How do we know the two systems preserve their own properties?
  - How can we re-tune them to ensure properties at both levels?



48

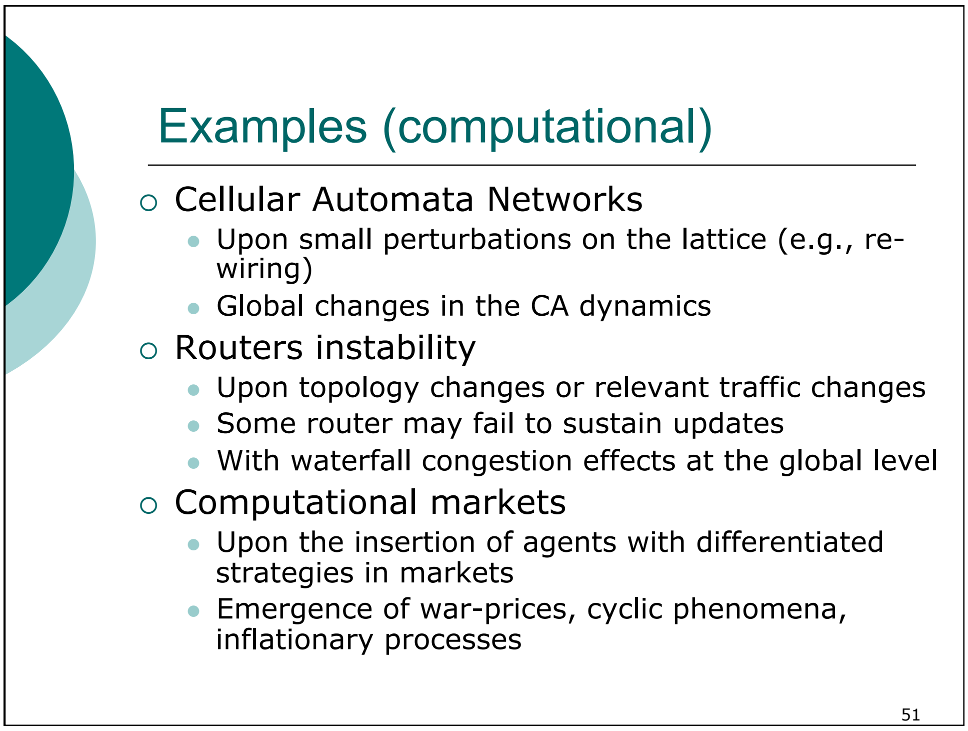
The Problem of Emergent, Unexpected Behaviors

- Very complex systems (i.e., macro-scale self-organizing systems), unlike simple ones
  - Exhibits non-linear relationships between structure and behavior
- Changes in structure can
  - Do nothing or
  - Dramatically affect the behavior



Examples (non computational)

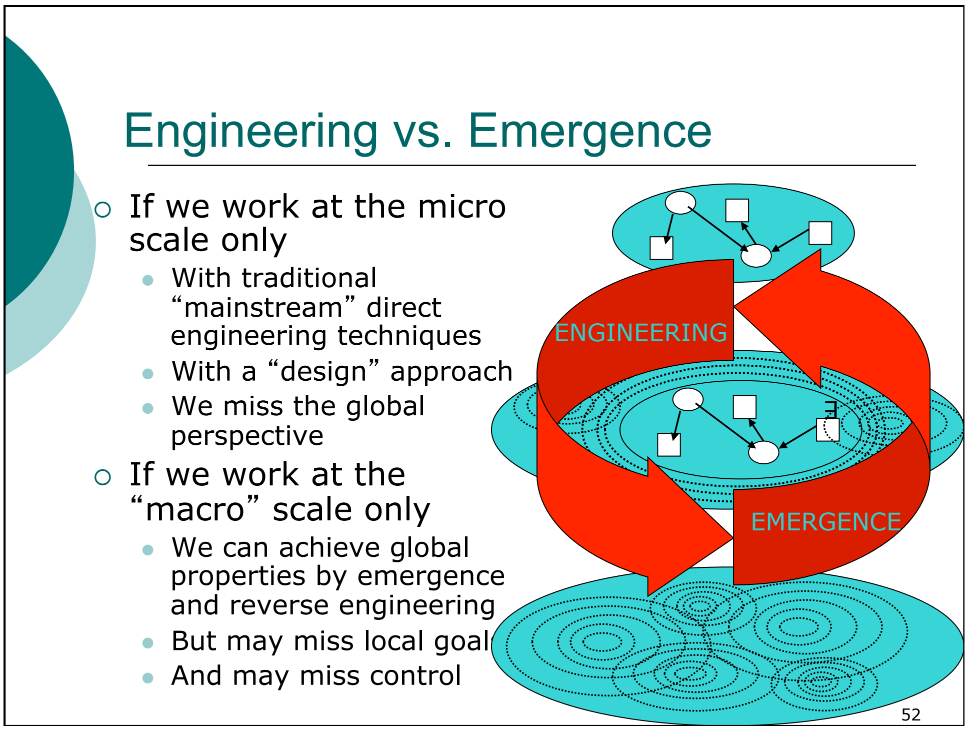
- Traffic Management
  - We know well how a roundabout (a sharp example of self-organizing system) works *per se*
  - But what about its impact in a complex network of streets?
- Ecology
  - We may know a lot about a specific ecosystems and about specific species
  - But what about the introduction of a new species into an ecosystem?
  - The Internet



Examples (computational)

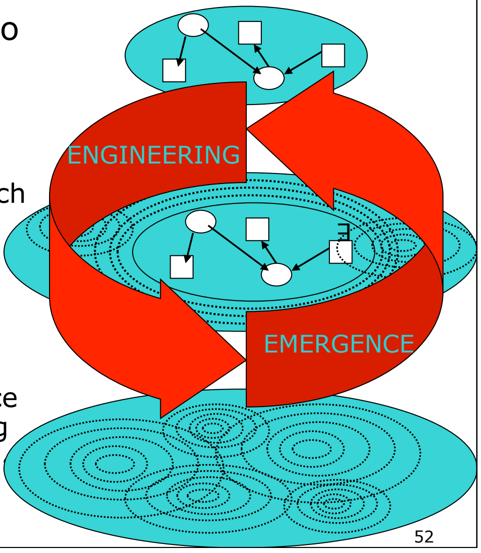
- Cellular Automata Networks
  - Upon small perturbations on the lattice (e.g., re-wiring)
  - Global changes in the CA dynamics
- Routers instability
  - Upon topology changes or relevant traffic changes
  - Some router may fail to sustain updates
  - With waterfall congestion effects at the global level
- Computational markets
  - Upon the insertion of agents with differentiated strategies in markets
  - Emergence of war-prices, cyclic phenomena, inflationary processes

51



Engineering vs. Emergence

- If we work at the micro scale only
  - With traditional "mainstream" direct engineering techniques
  - With a "design" approach
  - We miss the global perspective
- If we work at the "macro" scale only
  - We can achieve global properties by emergence and reverse engineering
  - But may miss local goal
  - And may miss control



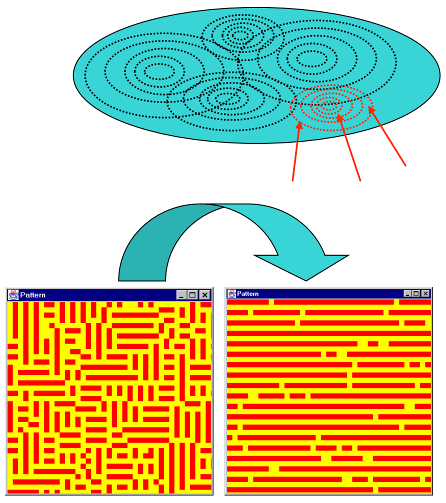
52

But why this is important?

- It is not only the recognition of a basic need
  - Most real cases face such issues
- More important, knowing what happens when we have a new system into an existing one
  - Can form the basis for new forms of **decentralized control**
  - And a new approach to engineer emergent systems
- Have a complex self-org macro system and
  - Know what happens when acting on it
  - Knowing how to enforce a specific behavior on it
- Shifting from
  - Local Rules → Hopefully Useful Global Behavior (pure macro-scale reverse engineering approach) TO
  - Local Rules + Decentralized Control → Engineered Purposeful Emergence

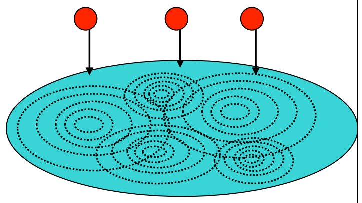
Decentralized Control via External Perturbation

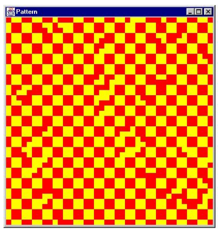
- Try to
  - Disturb the system where and how you can
  - Attempting to identify perturbation patterns that
  - Affect as needed the emergent system behavior
- Example with Cellular Automata
  - Introduce a moderate degree of stochastic behavior in cells
  - And have global (otherwise non emerging) patterns emerge



Decentralized Control via Components Injection

- Try to
  - Inject new components/ subsystem
  - That interact with the systems so as to
  - Affect as needed the emergent system behavior
- Example with Cellular Automata
  - Inject a few percentage of cells with modified rules
  - And have peculiar needed patterns (very unlikely attractors) emerge



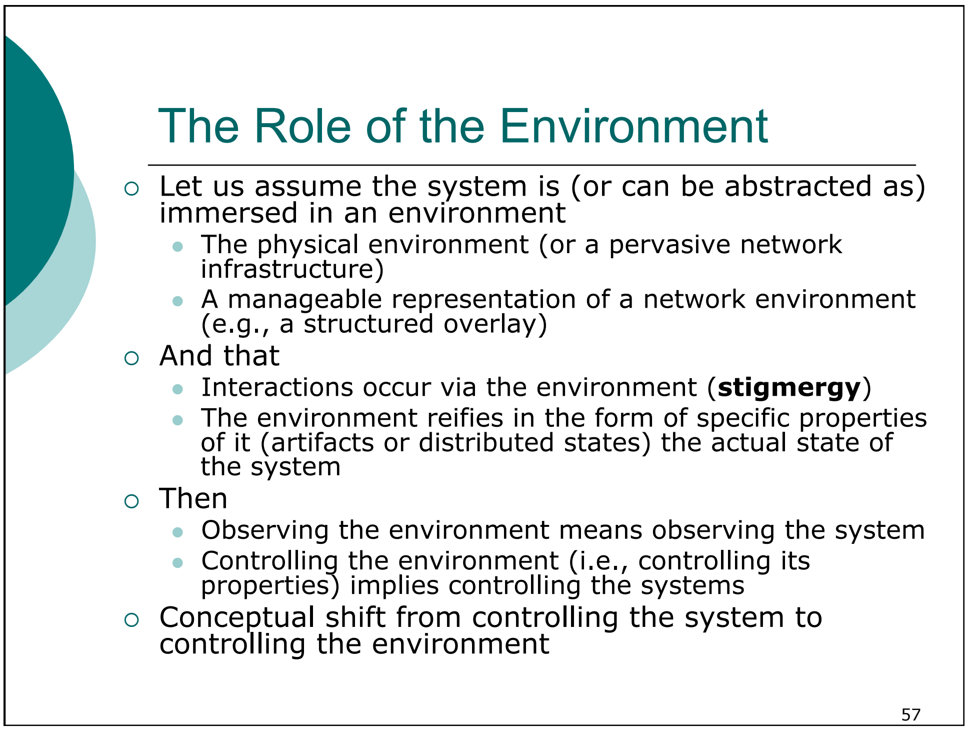


55

Engineering Emergence at the Meso Scale

- What is needed to advance knowledge in decentralized meso-scale control?
  - So as to produce a set of practical tools
  - Enabling the engineering and control of complex self-org systems
  - In a methodical and repeatable way?
- Conceptual advances in modeling
  - Discrete vs. Continuum Computing
  - Logics vs. Physics
  - Genotypes vs. Phenotypes
  - Design vs. Intention
  - Topology vs. Dynamics
- Or maybe the adoption of more usable abstractions other than those of local rules and local interactions?

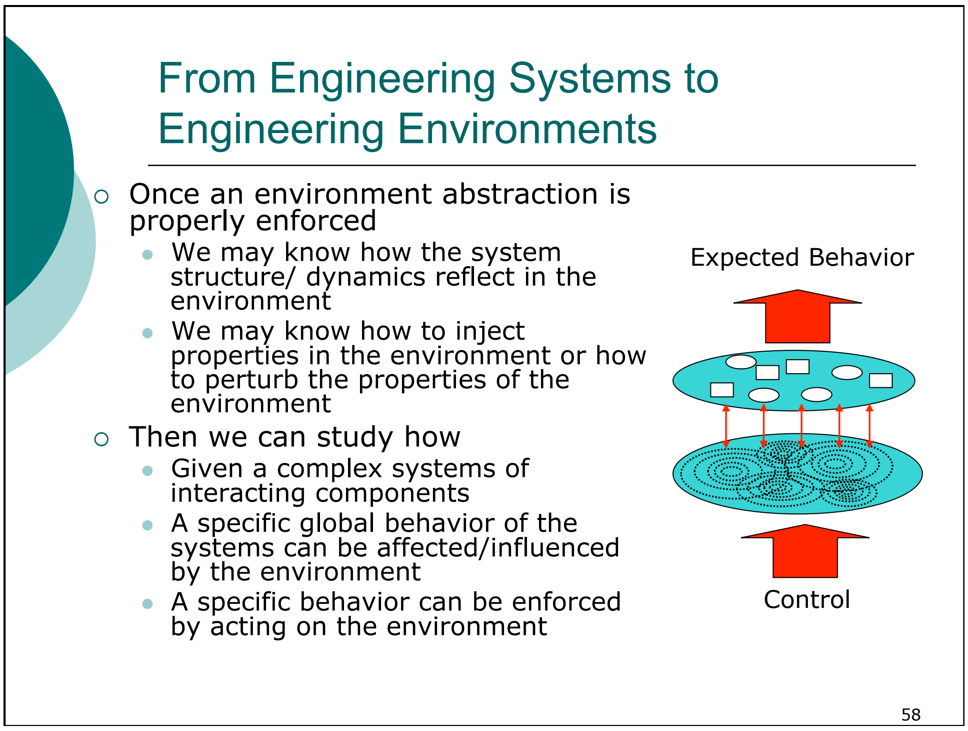
56



The Role of the Environment

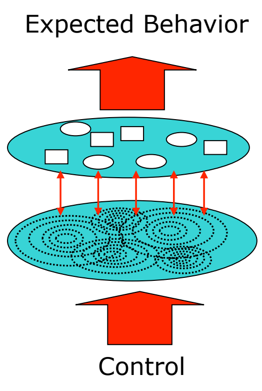
- Let us assume the system is (or can be abstracted as) immersed in an environment
  - The physical environment (or a pervasive network infrastructure)
  - A manageable representation of a network environment (e.g., a structured overlay)
- And that
  - Interactions occur via the environment (**stigmergy**)
  - The environment reifies in the form of specific properties of it (artifacts or distributed states) the actual state of the system
- Then
  - Observing the environment means observing the system
  - Controlling the environment (i.e., controlling its properties) implies controlling the systems
- Conceptual shift from controlling the system to controlling the environment

57

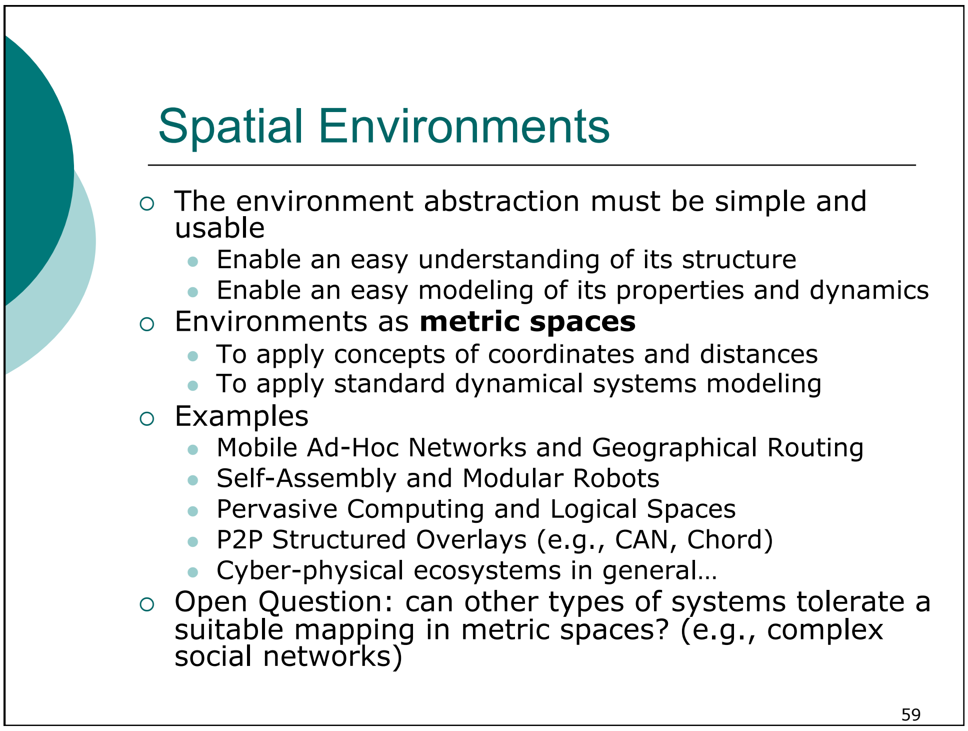


From Engineering Systems to Engineering Environments

- Once an environment abstraction is properly enforced
  - We may know how the system structure/ dynamics reflect in the environment
  - We may know how to inject properties in the environment or how to perturb the properties of the environment
- Then we can study how
  - Given a complex systems of interacting components
  - A specific global behavior of the systems can be affected/influenced by the environment
  - A specific behavior can be enforced by acting on the environment



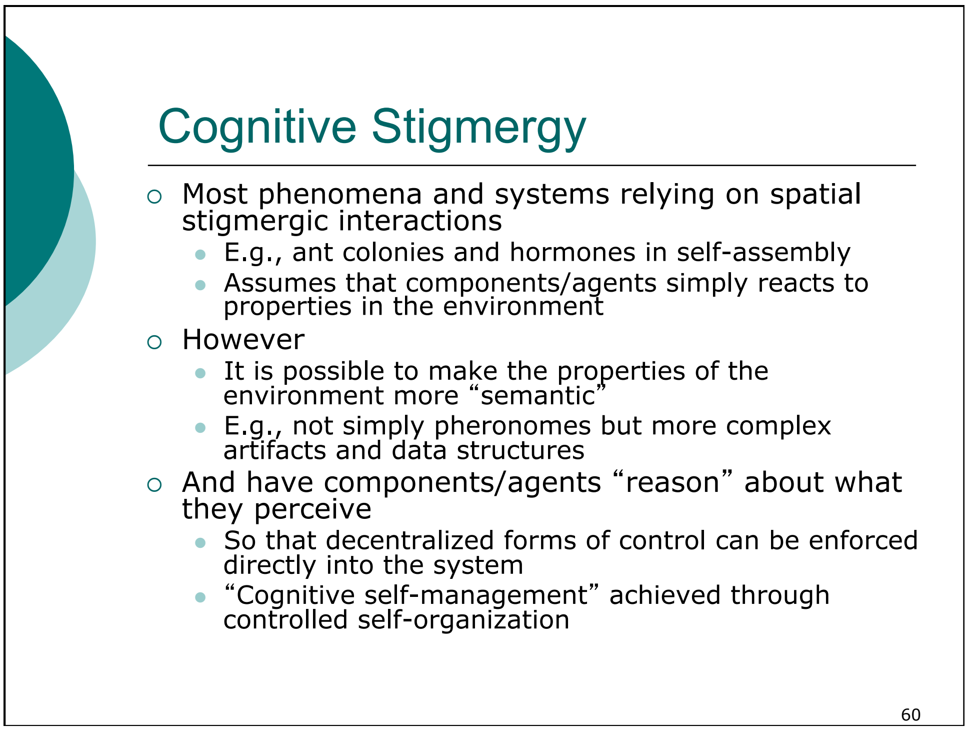
58



Spatial Environments

- The environment abstraction must be simple and usable
  - Enable an easy understanding of its structure
  - Enable an easy modeling of its properties and dynamics
- Environments as **metric spaces**
  - To apply concepts of coordinates and distances
  - To apply standard dynamical systems modeling
- Examples
  - Mobile Ad-Hoc Networks and Geographical Routing
  - Self-Assembly and Modular Robots
  - Pervasive Computing and Logical Spaces
  - P2P Structured Overlays (e.g., CAN, Chord)
  - Cyber-physical ecosystems in general…
- Open Question: can other types of systems tolerate a suitable mapping in metric spaces? (e.g., complex social networks)

59



Cognitive Stigmergy

- Most phenomena and systems relying on spatial stigmergic interactions
  - E.g., ant colonies and hormones in self-assembly
  - Assumes that components/agents simply reacts to properties in the environment
- However
  - It is possible to make the properties of the environment more “semantic”
  - E.g., not simply pheronomes but more complex artifacts and data structures
- And have components/agents “reason” about what they perceive
  - So that decentralized forms of control can be enforced directly into the system
  - “Cognitive self-management” achieved through controlled self-organization

60

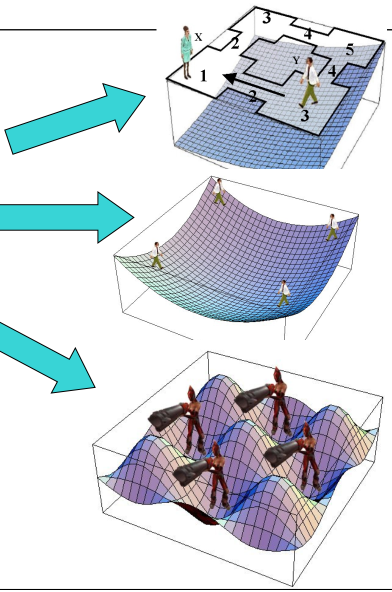
Engineering Emergence in TOTA

- Micro-scale
  - Exploit fields as a sort of distributed shared memory for contextual interactions
  - Tolerating network and environmental dynamics
- Macro-scale
  - Exploit fields to re-produce known phenomena of self-organization
  - Or to invent new (we can invent our own laws for field propagation)
- Meso-scale
  - When a specific (micro) field-based application is immersed in a macro-scale scenario
  - Proper combination of fields can accommodate both
    - The needs of the micro-scale
    - The needs of the macro-scale
  - In any case, new fields can be injected for the goal of enforcing the needed controls

61

Engineering Emergence in TOTA: An Example (1)

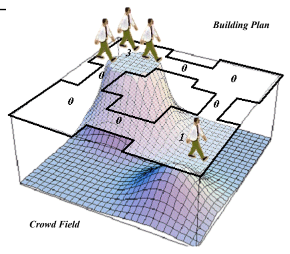
- **Micro-scale**
- Users visiting a museum
  - Where are you? → PRESENCE fields
- Meeting by tourists
  - Tourists following each others' PRESENCE fields
- Flocking by museum guards
  - As in the flocking example
- All of these realized by application-specific and independent fields

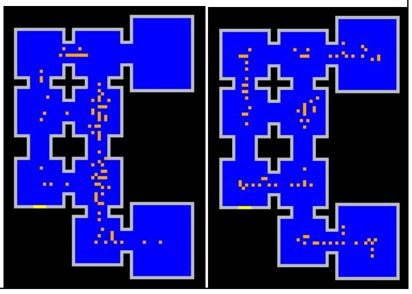


62

Engineering Emergence in TOTA: An Example (2)

- **Macro-scale**
- Diffusive Load Balancing of Crowd
  - Global diffusion of Presence fields
  - Weighted with data expressing room capacity
- Users behavior
  - Can follow suggestions
  - Can analyze what's happening (cognition!)
- Overall good load balancing even if a limited percentage of users follows the fields suggestions





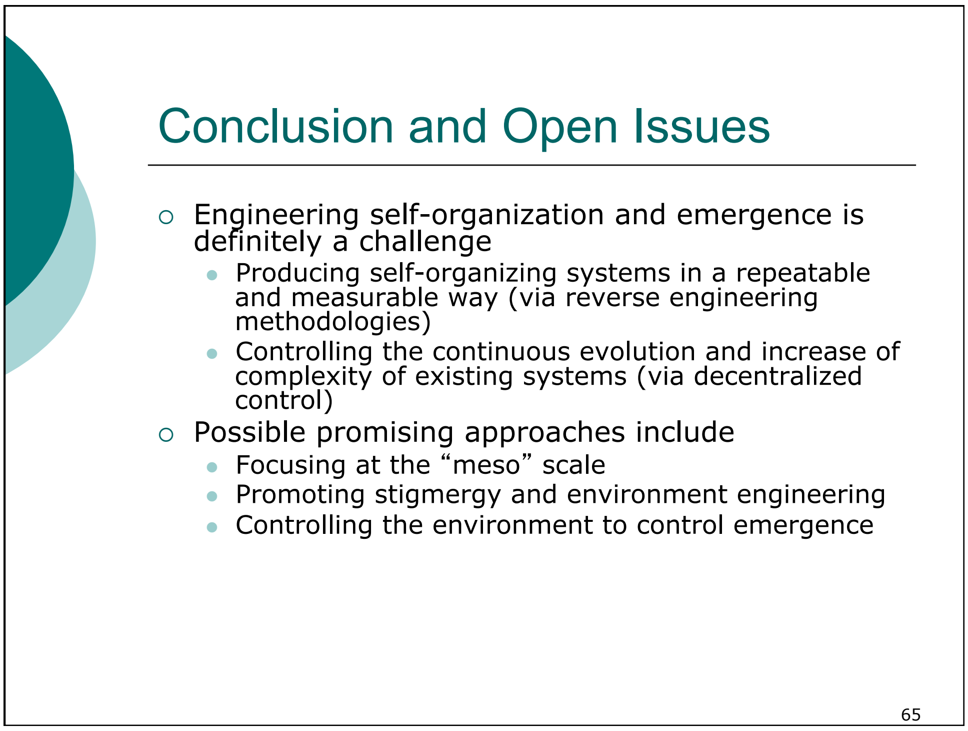
Engineering Emergence in TOTA: An Example (3)

- **Meso-scale**
  - Flocking + Load Balancing
  - How can we conciliate?
- Approach:
  - Have flocking agent perceive (and react upon) the following field:

$$Coord\_Field_i(x,y,t) = Flock\_Field_i(x,y,t) + \mu \cdot LB\_Field(x,y,t)$$

  - And tune μ (shape perceived fields) as needed

- μ=0
  - Ignore load balancing, and do pure flocking
  - Small decrease in load balancing quality
- μ>0
  - Flock accounting for crowd
  - Decrease in the accuracy of the flock formation
- In any case, each single agent can "see" the individual fields and take actions accordingly



Conclusion and Open Issues

- Engineering self-organization and emergence is definitely a challenge
  - Producing self-organizing systems in a repeatable and measurable way (via reverse engineering methodologies)
  - Controlling the continuous evolution and increase of complexity of existing systems (via decentralized control)
- Possible promising approaches include
  - Focusing at the “meso” scale
  - Promoting stigmergy and environment engineering
  - Controlling the environment to control emergence

65