Complex Adaptive Systems:
an Introduction

Franco Zambonelli

About Me…

- Professor in Computer Science
  - Università di Modena e Reggio Emilia
  - franco.zambonelli@unimore.it
- Expertise in distributed systems engineering
  - And in particular in the application of complex and adaptive systems solutions to distributed computing
  - 3 books, about 150 scientific papers
  - International awards
  - Coordinator of several EU projects on the theme
- Teaching
  - Complex adaptive systems
  - Operating systems and Java programming

About this Course

- Trying to give answers to the following questions:
  - What are complex adaptive systems (CAS)?
  - How do they work?
  - Why are they important for modern ICT systems?
  - Technologies that now more closely approach CAS (multiagent systems and swarm intelligence)
- And eventually
  - Discover how the lessons of CAS can help us better understanding modern ICT system
  - Learn how to engineer innovative solutions inspired by CAS

Outline of Today

- What are CAS? That is…
- What are Systems
  - Components, interactions, dynamics
- What is Adaptivity
  - Openness, situatedness, context-awareness
- What is Complexity
  - Complicated vs. complex Systems
  - Complexity vs. chaos
  - Relations with natural systems
- Why CAS in Modern Distributed Systems are?
  - Characteristics of modern distributed systems
  - Needs for CAS and Self-organization

- The Example of Cellular Automata
- Complex Networks and Social Networks

What are Complex Adaptive Systems?

- Let's understand this by characterizing the concepts of
  - ***System***
  - ***Adaptivity***
  - ***Complexity***
- Contextualize this to distributed computational systems
- And exemplify with the CA example

Part 1

- What is a System?

What are “Systems”?

- A System is an ensemble of (situated) individual entities interacting with each other
  - Individual entities:
    - atoms, cells, organs, animals, trees, humans
    - Chips, computers, LANs, sensors robots
  - Interacting
    - Physical forces, protein diffusion, gestures, words
    - Electric Signals, packets, IP datagrams, radio signals
  - Situated
    - Existing in an “environment” (see next)

Modeling Individuals in a System

- In general, an entity in a system is characterized by:
- A state
  - The dimensions fully characterizing the current situation of the individual
    - E.g., position and speed (X,dX/dt) for a physical object
    - E.g., the value of its instance variables for a software object
    - E.g., the "mood" for a human
- Transition rules
  - What events in the system make an individual change its state
    - i.e., what interactions
    - E.g., a gravitational force, the finding of food, a conversation…
- A Behavior
  - What the individual do in the system, how it affect the system
    - E.g., spreading of a gravitational force, eating food, talking, or simply occupy space
    - An object invoking methods of other objects
    - A Servlet changing a database
- A Context (a "position" in the environment)
  - Sometimes, it is useful to model a system and its individual as being "situated" in some environment
    - E.g., a topological space-time universe, a landscape, a room,
    - A servlet context
  - Other times, the environment in turn is modeled in terms of individuals
    - E.g., the space time being made of gravitons, a landscape being made of individual trees, grass, mote, etc.
    - In an OO program, everything is an object

Modeling Interactions in a System

- An interaction is any event in a systems that
  - Caused by an individual in a system
  - Affect that state or one or several individual in the system
- Types of interactions
  - Direct ("Communication")
    - Two particles colliding with each other
    - Two humans talking with each other
    - A client and a server exchanging TCP packets
  - Indirect ("Stigmergic" or mediated by the environment)
    - The modification of the speed of a particles caused by the gravitational field of another particles
    - The spread of pheromones by ants
    - Playing Chess
    - Two servlets accessing the same Attribute
- A note:
  - Sometimes, mediated interactions are not considered (i.e., fields in modern physics are modeled as particles!)
  - But this may be very important in distributed computational systems!!!

Distributed Computational Systems

- A Single computer is a system
  - Many components, to be orchestrated
  - By the operating system
  - Via coordination of access to common resources (stigmergic)
- Computer networks, in general…
  - Individual entities: computers
  - Networks: LAN + Internet
  - Interactions:
    - Direct: Message-passing, Client-server, Events
    - Stigmergic: Access to common databases and resources
- But with a very broad meaning
  - Computers: Supercomputers, PC, PDA, Mobile Phones, embedded computers
  - Networks: Wires, Wireless, GPRS, UMTS, Bluetooth, P2P, The Web, The overlying social networks
  - Interactions: Access to common memory, Access to common artifcats, RF-ID Tags, Broadcasting, Digital Pheromones, Digital fields
  - Will analyse all of these….

Discrete vs. Continuous & Synchronous vs. Asynch. Systems

- The modeling of a system may imply
  - Continuous vs. Discrete state
  - Continuous vs. Discrete time
- Global vs. Local Time
  - Global: all the individual of the system have the same clock, and thus perform state transition at the same time
    - ***Synchronous***: when they perform state transition at the same time or when there is a sort of global controller for transitions
  - All the individual have individual clocks
    - ***Asynchronous***: when they do not obey to global clock rules to perform state transitions, and there is not global controller
- Summarizing:
  - **Physics** is discrete in state, continuous in time, and asynchronous
  - **Ecology** is continuous in state, continuous in time, and asynchronous
  - **Computing** is discrete in time, discrete in time and synchronous
  - **Distributed computing** is discrete in time, discrete in time and asynchronous

Local vs. Global System State

- The collection of all states of all individual of a systems (i.e., the **local states**), plus the state of the environment, if needed, determines:
  - The **global state** of the system
- However, in many cases, the global state is represented by more "synthetic" indicators, Because
  - It give a synthetic clue of what is happening in the system
  - Sometimes, it is impossible to determine the global state in terms of all local states
- Examples
  - "The current average speed on the highway is 45mph"
  - "The DISMI network has a throughput of 345kbs"
  - "The unbalance in computer load is 45%"
  - The state of a gas is measured by P & T, because it is impossible to measure the positions and speeds of all its atoms

System Dynamics

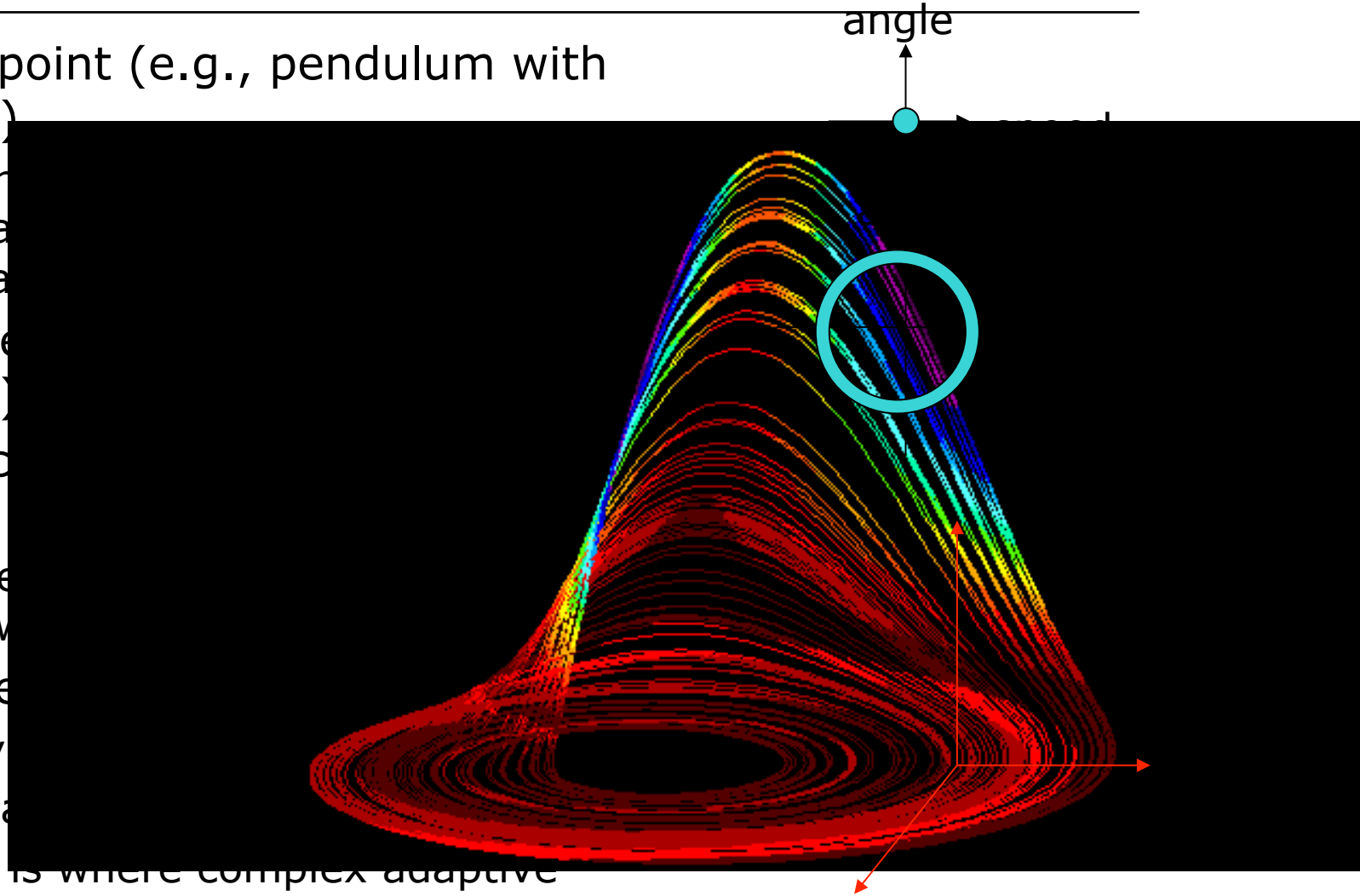
- A system evolves in time
  - Via the **local** state transition (local state changes) of its individuals
  - Affecting the **global** system state
- The study of system dynamics imply
- Determining the evolution of the system
  - Given an initial state
  - E.g., will the system reach a global equilibrium? Of what type? From what initial states will reach what equilibrium state? Will it continue evolving indefinitely? What happens when we perturb the system from equilibrium?

The “Phase Space” of a System

- When studying system dynamics, one
  - Does not refer to the evolution in state of all its system components
  - But rather to the evolution of some of its relevant global state indicators
    - E.g., the evolution of a gas is described in terms of how its temperature and pression change in time
  - This is the so called “trajectory in the state space”
    - The dynamic evolution of the system as a point moving in a n-dimensional diagram, starting from an initial point
    - Axis represent the relevant indicators of the global system state
    - And in which time passes as the point moves…
- Typically, the trajectory tend to be “attracted” in specific regions of the diagram
    - “**Attractors**” of the systems (single points for static equilibrium, circles for periodic behaviours, or more complex manifold)
    - **Attractors express stability**: once the system reaches an attractor, it will stay there undefinitely, until perturbed

Movements and Attractors in the Phase Space

- Single point (e.g., pendulum with friction)
  - From
    dyna
    towa
- Cycle (e
  friction)
- Chaotic
  gas)
  - Move
    the w
- Strange
  - Very
  - So ca
  - This is where complex adaptive systems stand…



Part 2

- What is an Adaptive System?

What is Adaptivity?

- The capability of a system and/or of its individuals of tuning its behavior to contingent situations
- Adaptivity implies that the system is both **flexible** and **robust**
  - Re-organizes itself in response to stimuli
  - Preserves the same overall dynamics
  - Preserves (some of) its specific characteristics independently of external contingencies/stimuli
- What are these contingencies requiring adaptivity?
  - **Openness** → the boundaries of a system are open
  - **Environmental Dynamics** → the environment may be dynamic and impredictable
  - **Structural dynamics** → the components can be themselves dynamic
- In general, one or more of these sources of dynamicity affect all modern distributed computing systems (details later)

Adaptive vs. Non Adaptive Systems

- Adaptivity in response to openness and environmental perturbations/dynamics implies
  - **Flexible response**: the system adapt its overall dynamics in response to stimuli
  - **Robust behavior**: at the same time, the system preserve its overall behavior, or at least some of its relevant characteristics

- However, please note that adaptivity is not an absolute concept
  - It depends on what we are interested in in a system
  - It sometimes depends on the eye of the observer and on the level of observation of the system

Earth vs. Heart

- The Earth mechanical dynamics
  - Non adaptive: a perturbation (e.g. a big earthquake) change its rotation period permanently
- The Earth climate
  - Very Adaptive: stable over very long periods despite pollution and sun storms
  - But non adaptive from the viewpoint of LA's inhabitants ☺
- The Heart dynamics
  - Very adaptive: fast response to stimuli (pump more blood on need), robust behavior (restore its normal flow a few minutes later)
- The Heart structure
  - Non adaptive: once damaged, it cannot repair itself (e.g., think at heart attacks!)

Adaptive Individuals vs. Adaptive Systems

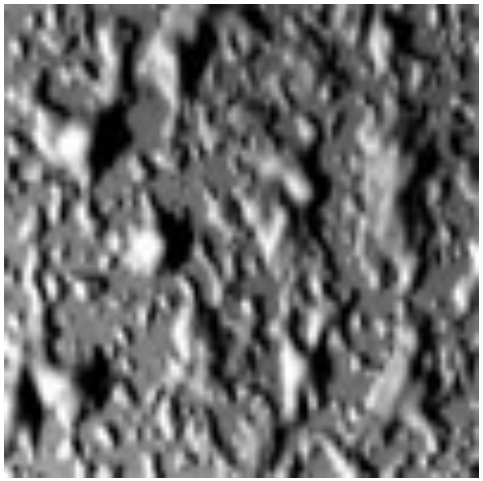
- Sometimes, individual may be adaptive
  - Cats can live anywhere and adapt their food habits accordingly
  - Chameleons change color depending on the environment

- Making the whole system consequently adaptive
  - Cat colonies live and prosper anywhere on the planet

- But sometimes individual capabilities the system overall may not
  - Chameleons communities strongly endangered by pollution and (guess what) by cats ☹
  - The "dynamics" of the chameleons system outperforms the dynamics of the individuals

Non Adaptive Individual vs. Adaptive Systems

- In several cases, ensembles of very simple, purely "reactive" components, may overall lead to very adaptive systems
  - The cell of the heart are not per se adaptive or "smart", nevertheless...
  - The simple unicellular components of the Dictyostelium group together to hunt



- In these cases, adaptivity is a capability that "emerges" from the system and from its interactions
  - Induced by the specific dynamical behavior of the system, as determines by the simple behavior of for the very fact of being a system with a specific dynamics
- Here, adaptivity is a consequence of complexity, and of the system being "at the edges of chaos"

What are we Interested In?

- All the discusses classes of adaptivity are of great interest to us (and will be analyzed)
  - "Smart" (hardware/software) components able to exhibit adaptivity as individuals (**agents**)
  - "Smart" components able to exhibit adaptivity in multicomponent systems (**multiagent systems**)
  - "Stupid" components able to exhibit collective adaptivity, i.e., they are collectively smart (**cellular automata, complex and social networks, swarm intelligent systems**)

Adaptive Computing Systems

- Let us now turn our attention to various computational systems, and see
  - why they have to be adaptive
  - how they can achieve adaptivity

Computational Openness and Network Dynamics

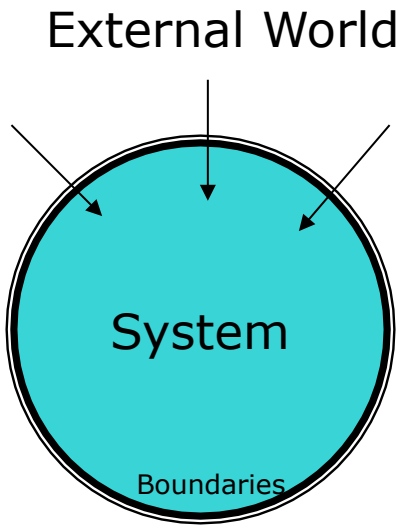
- Openness is a general characteristics of modern decentralized distributed systems
  - **Decentralization**: may stakeholders can add computers and components (e.g., Web pages and components). No central control.
  - **Mobility**: users/components move while being connected to the network (PDA, mobile phones, vehicles) so that the structure of the system continously change
- Also, consumer and embedded computing systems
  - **Ephemeral**: Lead overall to very dynamic systems, with nodes coming on and away at any time

Situatedness in Physical Environments

- Modern computing systems are more and more made able to interact with the physical world
  - **Localization mechanism**, GPS
  - **Sensors and actuators** to understand and affect what is happening in the physical world
  - And this is definitely the case for mission critical systems
- Therefore, the physical world is becoming a central part in distributed computing
  - Location aware computations, that adapt their behavior depending on where the user is
    - A Service that adapt the information provided to vehicles depending on their location
  - Situation-aware services:
    - a smart user interface that adapt its lightening depending on the external visibility
    - A security sensor that recognize illegal activities in a park and alert the police
  - More generally, "ambient intelligent" systems

Dealing with Openness and Dynamics

- Let's "model" explicitly
  - What it is inside the system
  - What it outside (entities and perturbation forces)
- Identify and model the "interactions across boundaries"
  - What comes in goes out
  - How the environment can change
  - Sources of structural dynamics
- The boundaries of the system somewhat identify the "***environment***" or (which is the same) the "***context***" in which the system situated and its dynamics
  - The environment is an abstraction!!!
  - Not always easy to identify

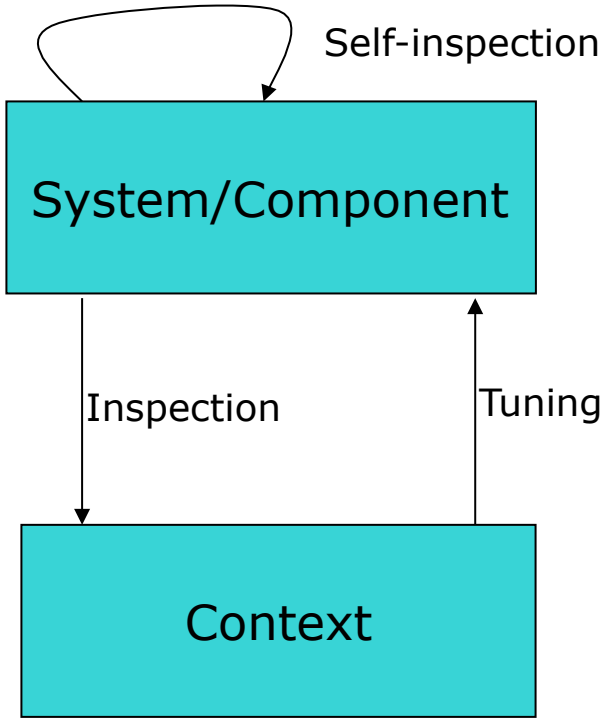


Context-awareness

- More in general, for computing to be adaptive it has to be
  - **Context-Aware and context-adaptive**
- Which implies
  - The system or its components must be able to recognize the context in which they situates
    - A computational context (the set of file, services, communication and computation resources, as e.g., in servlet context)
    - A physical context (the real-world)
  - Must recognize changes in context
  - Must adapt their behavior to changes in such context
- This can be achieved either via the adaptivity capability of the individuals, or via the adaptive capability of the system as a whole
- So, it is rather clear that adaptivity may be strictly related to the overall dynamics and complexity of a system…

Adaptivity via Context-awareness

- In general, it is achieved via
  - Inspection of the context
  - Tuning of behavior depending on context
- This can occur at the level of individuals or at the level of the system
  - And it can include "self-inspection"
  - the systems and its component are by definition part of the context
- It can be supported by proper technologies to help recognizing context and properties of the system
  - "middleware" in the end is something that enables context-awareness and adaptivity…
- **Agents** (discussed in later lessons) are **inherently context-aware** components
- But here in this course we are also interested in something more: **adaptivity via dynamic self-organization in complex systems!**



Part 3

- What is a Complex System?

What is a Complex System?

- A system is complex when it is hard to infer its dynamics from the behavior and state of its individual components → difficult or impossible to detect its evolution in the phase space
- There are various reasons for this:
  - **Local interactions**: usually, interactions occur among a limited set of "neighbor" components, so that their global transitive effect on the global system state if hard to be elaborated
  - **Non linear dynamics**: e.g., x(t+1) = x(t)(1-x(t))
  - And **feedbacks in interactions**: A→B, B→A,
    - Both typically leading to equations difficult to solve, and to complex phase space topologies with multiple complex attractors. For instance, a system can evolve in very differentiated ways even from very close initial points in the phase space ("butterfly effect").
  - **Openness and Environmental Dynamics**: these prevent
    - A definite modeling of the system, as the system keeps on changing
    - Any predictive modeling of the behavior. The system is always kept out of equilibrium, so that it is difficult to simply trying to see how a system will evolve in the phase state diagram. Probabilistic models must be considered

Complex vs. Complicated Systems

- Clearly, in nature and engineering there are many systems which are very complicated
  - Many components, many interactions
- However, a system which is only complicated and not complex typically has
  - Linear interactions, facilitating modeling and integration of dynamical equations
  - No feedback loops, avoiding the butterfly effect
  - A single or a limited set of stable equilibrium points or limit cycles
  - Contiguous points in the phase space attracts towards the same attractors
  - It is closed, or has well defined boundaries with predictable interactions across boundaries. So it can be treated determi
- Example:
  - The engine of a car can be very complicated, but it is not complex
    - A turbo engine can become complex (the feedback cycle induced by the turbo injector)
    - A badly cooled system can become complex, due to the thermodynamic interactions with the external world
  - A microprocessor, even with billion transistors, it typically only "complicated" → very linear and clean transition rules

Complex Distributed Computational Systems

- Most modern distributed computational systems are indeed complex
- They are open and situated, which is per se an important driving force for complexity
  - We have analyzed this w.r.t. adaptivity
- In addition, they are often based on local interactions
  - E.g., P2P system, mobile systems
- Interactions are not linear
  - E.g., in the Web, the time to get a service grows more than linearly with the number of concurrent requests
  - Effects such as TCP timeouts introduce non-linearity
  - The behavior of humans is greatly non-linear, necessarily reflecting the network system
- And they contain feedback loops
  - The network has loops
  - The more we wait for a service, the more we keep on polling (reinforcement feedback)
  - In P2P network, the more the users the more the system grows and get used

Complexity vs. Size

- Size (the number of components in a system) per se, is not a key reason for complexity
  - There are several huge systems which are simple and linear
  - Control hierarchies, engines, microprocessors
- However, size may be a necessary pre-condition for complexity
  - E.g., Two gravitational masses cannot be a complex system, Three masses are (the three body problem), Asteroids in the Kuiper belt are subject to complex movements
- For Distributed Computational Systems
  - The size may be so large that complexity can hide in them
  - In general, size (together with decentralization) prevents a complete micro-level knowledge of its local components state, e.g., it is nearly impossible to know exactly all the peers in Kazaa

Equilibrium, Chaos, and the Edges of Chaos

- It is important to note that a complex system it not necessarily chaotic
- Rather, many types of complex systems found in nature are at the "edge" of order and chaos: critical order
  - **Order**: global equilibrium of the system, uniformity
  - **Chaos**: disordered out-of-equilibrium behavior, unorganized non-uniformity
  - **Edges**: out-of-equilibrium, organized and structured behavior, although not exactly predictable
- Clearly, changing some working parameter of the system may lead it to become ordered or chaotic…
- Please note that the "chaos" word often refer to system that are subject to the butterfly effect (e.g., the Earth atmosphere).
  - However, the butterfly effect does not imply that the system is chaotic, but simply that
  - It can evolve in very different configurations starting from however close initial conditions

Complexity vs. Chaos: Examples

- A Dead Heart
  - Static equilibrium, total order
- A Health Heart
  - Global synchronization pattern (order!) of its cells, achieved via local interactions and feedbacks
  - However, the synch is never so exact, and may slightly vary in period
  - Edges of chaos…
- A Heart during an attack
  - Fibrillation, chaotic pulsing of its cells

Self-organization in Nature

- Many natural systems – even when composed of very simple or stupid elements, appears able to self-organize themselves

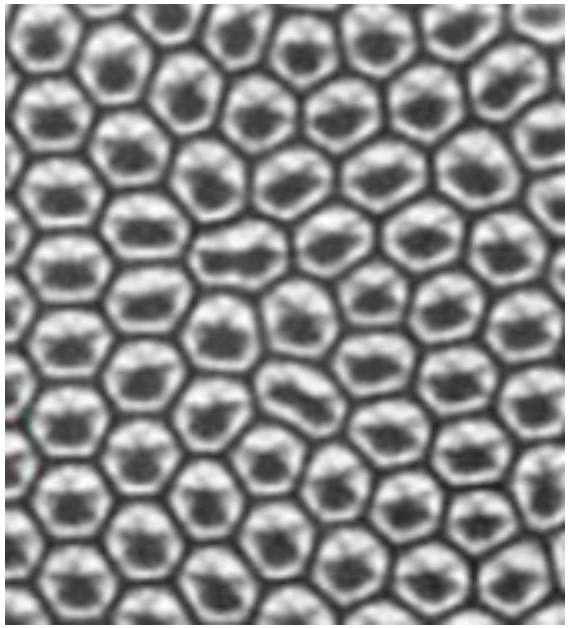
  "Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower level components of the system". [Deneubourg 1977]

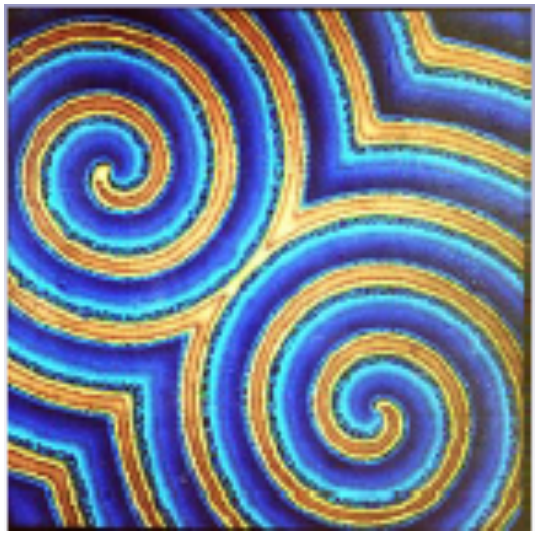
- Actually, these systems exhibit a behavior which is clearly at the edge between order and chaos
- In most cases, due to the presence of non-linearity and both positive and negative feedbacks
  - Positive feedbacks introduce amplifications and move the system out of equilibrium
  - Negative feedbacks self-regulate the system and avoid it to become chaotic

- The behavior is considered "emergent"

Examples of Self-organization in Nature (1)

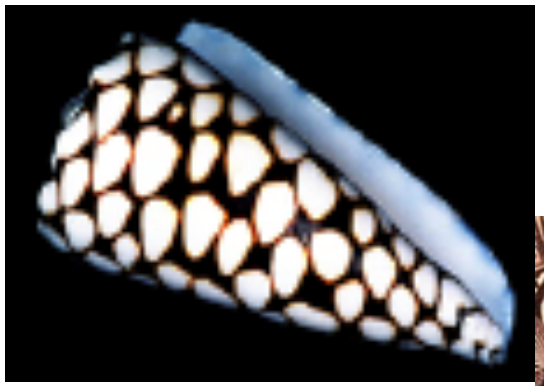
- Patterns in Physics





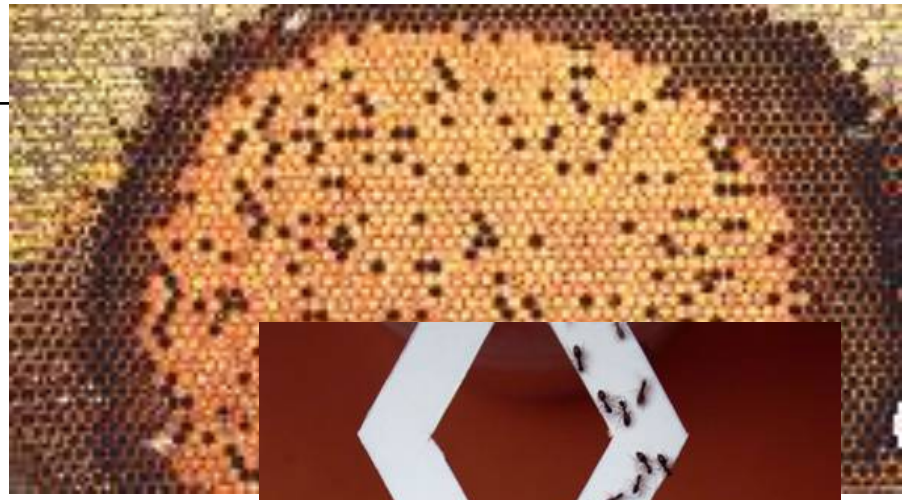
Examples of Self-organization in Nature (2)

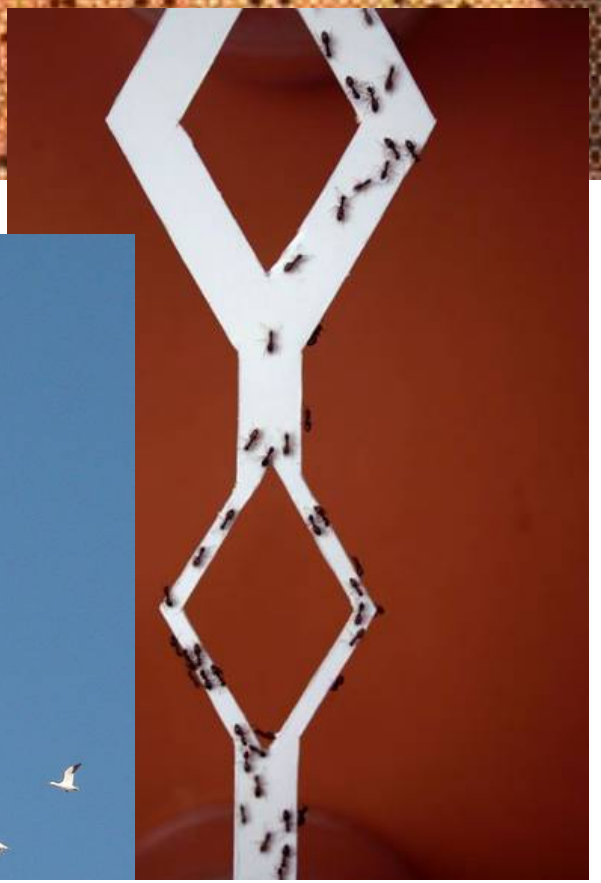
- Patterns in Skins and Shells



Examples of Self-organization in Nature (3)

- Dynamic Patterns in Group Behavior





Examples of Self-organization in Nature (4)

- Emergent Behaviors in Societies
  - Clapping synchronization
  - Mexican Wave ("La Ola")
  - Patterns of Walking in crowdy pathways
  - Patterns of epidemics
  - Patterns of acquaintances
- The structure of networks
  - Many networks of different kinds (nature, society, technology)
  - Tend to evolve such as to self-organize specific structure
  - Adaptive and robust w.r.t. to both structure and information propagation

Complexity vs. Adaptivity

- A self-organized system may be adaptive for a wide range of environmental perturbations
  - It can re-establish global patterns upon changed conditions
  - Typically not the same, but similar
  - It moves from an attractor to an equivalent one
  - The presence of negative feedbacks, provide for re-establishing order
  - Clearly, excessive perturbations may disrupt the system or move it to chaotic state
    - E.g., the human heart
- Also, openness is a limited problem
  - New components can get in the system
  - Without affecting significantly its overall dynamics

Part 4

- Why Exploiting CAS in Modern Distributed Systems?

What Characterize Modern Distributed Systems?

- Large, decentralized, situated and dynamics networks
  - Internet, Web, P2P, sensor networks, WiFi meshes
- High complexity of software and underlying hardware
  - very hard to be managed and engineered, due to size and non linear interactions
- Continuous updated and tuning needed
  - Dynamic changes in the network and in the system (openness), as well in the patterns of usage
  - Dynamic changes in the environment
- Impossibility of direct control and configuration
  - E.g., cannot control all the nodes of the Internet or of a P2P network
  - Cannot control and access all the sensors and embedded computers distributed in an environment
- Impossibility of direct maintenance
  - Cannot re-configure manually all the nodes
  - Application must not stop working and should preserve specific quality levels

Why Self Organization in Distributed Systems? Also…

- Commercial and Practical Reasons
- And even if we could control the configuration and the adaptation/maintenance of complex systems that would be
  - Economically unfeasible
    - Too high development and maintenance costs
  - Commercially unacceptable
    - Who wants a system which is always in need of configuration?
- Computing systems must become
  - Proactive, self-adapting and self-organizing with "humans out-of-the-loop" (Tennenhouse, 2001)
  - Autonomic, as if they were living organisms (Kephart, 2003)
  - Easy and painless to be used as "sprays" (Zambonelli, 2005)

What is Computational Self-Organization?

- Components/applications get deployed and:
  - They recognize who and where they are (w.r.t. the other components and the environment)
  - They identify their specific task in the network (according to the who and where)
  - They start working in cooperation with the other components to achieve their task
  - The global goal/configuration reached without supervision
  - Achiving in a spontaneous way **emergent, self-organized, coherent behaviors**
- Upon dynamic changes
  - They recognize such changes
  - And re-adapt to suit the new situation

So What?

- There is need for brand new:
  - Theoretical models
  - Algorithms
  - Middleware infrastructures
  - Programming abstractions
  - Methodologies and tools
- For building distributed computing systems that can
  - Show properties of natural complex adaptive systems
  - Rely on self-organization and emergent organization
  - Be robust to dynamics and be highly adaptive
- A great commercial advantage could be achieved in current applications
  - Faster deployment, reduced configuration and maintainance costs, natural evolution
- And a number of potential innovative applications could be conceived and deployed….