Chapter 5
The Making of Complex Systems

Introduction

In human factors, we learned during the last 3 decades that people commit various kinds of errors during operations of life-critical systems, and these errors are the cause of a majority of accidents. Therefore, we developed compensation responses for technology, organizations and people, including various kinds of defenses and resilient strategies, because relevant LCS properties were identified. It seems the problem is much deeper than the shallow solutions that have been found so far. Safety culture has to be based on a long-term educational process that should start at school from an early age. Today, such culture has to be reformatted into a complexity culture. What is a complex system? Why such a system could lead to catastrophic situations?

The natural world is full of complex systems, and scientists have studied such complexity for a long time, trying to identify persistent behavioral patterns, generic phenomena, relationships among components and so on (Mitchell 2009). Today, technological developments result in new kinds of complexity. Scientists developed **models** in order to deconstruct complexity, or more precisely see what was supposed to be complex in a different "simpler" way. For example, a long time ago people used to watch the sky as a complex set of stars. Copernic, Kepler and Galilée deconstructed such complexity by providing models based on two simple variables (i.e., distance and mass). Similarly, today's scientists develop models in order to describe what they observe (i.e., they rationalize complex datasets).

This book focuses on artifact design and use, referred to today as "Human-Centered Design". **Artifacts** (i.e., artificial things) differ from natural things because people build them. They can be simple, but we make them more complex by adding more properties. They can also be complex from the start because they are built to solve complex problems. Complex systems are usually defined as an integrated set of interconnected components. These systems can be aircraft, spacecraft, nuclear power plants, medical operating rooms, disaster-management centers and so on. These systems are inherently complex because they are cognitive prostheses (Ford et al. 1997). An aircraft is a cognitive prosthesis because it has been designed to supply people with flying capacity (remember that people do not fly naturally). We

G. A. Boy, *Orchestrating Human-Centered Design,*
DOI 10.1007/978-1-4471-4339-0_5, © Springer-Verlag London 2013

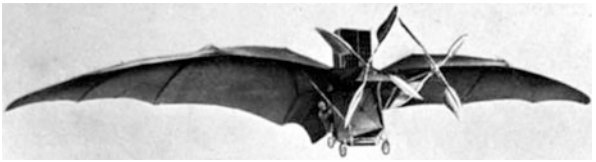

Fig. 5.1 Clement Ader's Eole. (Image is in the Public Domain, copyright free)

actually deconstructed the complexity of flying when we understood that flying was a matter of thrust and lift. Once humans had the right structural devices to ensure lift and the right propulsion to ensure thrust, we managed to fly. **Clement Ader** was one of the first to make this theory concrete. Ader built the first flying machine *Éole* (Fig. 5.1) that he attempted to fly on 9 October 1890 in the vicinity of Paris. He flew 50 m reaching a height of 20 cm, 13 years before the Wright Brothers (ASA 1994). Ader deconstructed the complexity of human flight by proving it was possible to physically stay in the air while propelled by an engine. Later on many other people designed, developed and used aircraft along these lines, refining the manned-flight concept. They mastered complexity of the manned-flight concept both functionally and structurally. Note that we incrementally learned from experience. Also, it is important to mention that aircraft improved over the years in the issues to safety, efficiency and comfort because these important topics were developed symbiotically with pilots. More importantly, the experimental test pilot (ETP) job emerged from a human-centered design approach, even if not formalized as it is today.

For a long time, systems were designed individually without much attention to their coupling to other systems, most importantly people. Their integration into an organizational environment was always the source of various surprises and discoveries, sometimes a few catastrophes. Adjustments were always necessary and often systems had to be redesigned to fit the reality of their actual use. This kind of process seems natural and acceptable during development phases, but is not acceptable once a system is delivered. This is a question of technological maturity that deals with intrinsic complexity, and maturity of practice that deals with extrinsic complexity. This distinction between intrinsic complexity and extrinsic complexity is important. **Intrinsic complexity** results from system architecture and internal relationships among its components. **Extrinsic complexity** results from the activity of the various agents involved in the use of the system (i.e., interaction between the system and its environment including its users). Of course, intrinsic complexity and extrinsic complexity are intimately related, but it is often much easier to study them separately and later investigate how they relate.

Perception of complexity is often a matter of **ignorance**. The more we know about something, the more we find it simple to understand and manipulate. For example, it might be very complex to find your way driving to a place that you never went before. You need a map, directions and most importantly you need to think and act appropriately. After you have gone to a place once, it becomes simpler to get there a second time. You have constructed a pattern (or model) for going to this place. You even refine it by optimizing time and distance. In fact, complexity was mainly in your ignorance of salient entities and relationships among them, which determine the relevant patterns. Discovering and learning patterns/models tends to deconstruct complexity.

In Chap. 2, we already saw that maturity of a product is strongly related to the quality of its high-level requirements. Initial conditions matter when we deal with complex systems. The famous meteorologist Edward Lorenz proposed a weather model that demonstrated we cannot predict precise weather forecasts more than a few days ahead, because of sensitivity to initial conditions (this sensitivity is called the Butterfly effect), but we can predict future weather patterns that are called chaotic attractors (Lorenz 1963). Let us take the 2011 Fukushima Daiichi disaster as an example.

The Fukushima Daiichi disaster was first a huge earthquake and tsunami that killed 20,000 people, and a subsequent nuclear catastrophe. Could this event have been predicted? Certainly not in the usual sense of linear mathematical prediction! However, there is geological evidence of six catastrophic tsunamis hitting the Sanriku coast within 6,000 years.[1] Among them, the 1896 Meiji Sanriku earthquake caused 22,000 casualties. Does this knowledge enable prediction? It was very difficult to predict when another earthquake/tsunami would happen when the decision was made to build a nuclear power plant in Fukushima Daiichi. However, it was clear that (1) the risk was very high due to the fact that this area of the world is very vulnerable regarding high-magnitude earthquakes and tsunamis, and (2) if an earthquake and/or tsunami did occur, there would be disastrous consequences. In addition, several decisions were made that did not take into account these two factors. Seismology deals with highly complex geological behaviors that are almost impossible to predict in terms of time of occurrence. However, there are highly possible areas of danger that can be considered as geographical attractors.

Back to design and engineering, following Lorenz's complexity approach, an end product can be very sensitive to high-level requirements since the processes and organization that contribute to the making of the product are complex. However, it would be possible to predict relevant attractors that will shape the main characteristics of the product. For example, it is now clear that taking a technology-centered approach will lead to the development of user interfaces that attempt to compensate and ultimately hide intrinsic complexity, and sometimes force users to adapt to the system (the inside-out approach to engineering). Conversely, taking a human-centered approach based on extrinsic complexity from the start will lead to an integrated symbiotic human-machine system in the end (the outside-in approach to design).

The inside-out approach has been used for a long time because engineered systems were less complex than the ones we know today. Therefore, it was easy in the past to design a user interface because the number of variables and parameters was reasonably small. Today, this number has become huge, and designers have to make difficult choices in the design of user interfaces. What is important to show? What should be controlled? Layers and layers of software have been developed to take into account safety, efficiency and comfort at the same time, increasing both intrinsic and extrinsic complexity. Consequently, systems have their own behaviors that people need to perceive, understand, consider and react to appropriately. We have moved into a human-system interactive world that cannot be considered

[1] http://en.wikipedia.org/wiki/Seismicity_of_the_Sanriku_coast.

as a single human operator using a machine, but as a multi-agent interactive environment. This is why the classical **positivist approach**[2] is no longer sufficient, often ineffective and inappropriate, so we prefer to use more **phenomenological approaches** to design and development. The positivism-phenomenology distinction will be emphasized in this chapter. In particular, a complex system is an articulated set of dedicated sub-systems that induces emerging phenomena, typically unknown at design time. Good human-centered design should focus on discovery of emerging phenomena.

Therefore, the **systems-of-systems** (SoS) approach is preferred to the commonly-used approach of designing systems individually and then integrating them in the end. Without designing the overall architecture and enabling its functionalities and behaviors, it is not possible to assess its extrinsic complexity (nor its intrinsic complexity). Various definitions of the SoS concept have been offered by many authors (Jamshidi 2005). It requires more work to get a workable definition. It originated in the defense sector (Luzeau and Ruault 2008). Today, we cannot think of air traffic management of the future without stating it in terms of systems of systems. Other examples are the Internet, intelligent transport systems, and enterprise information networks (or Intranets). It becomes obvious that this notion of systems of systems now integrates the distinction of intrinsic and extrinsic complexity.

Since systems of systems are made of humans and systems, original natural systems tend to become artificial systems. Many examples have emerged from this evolution such as genetically modified organisms, integrated prostheses, and so on. Technology is now part of our lives and needs to be investigated correctly if we want to take care of Human Kind and the Earth. This is why improving our understanding of complexity is very important when we, Humans, attempt to modify natural complexity. Consequently, **life-critical systems** need to be better investigated and understood in order to find the right mix between humans and machines.

Non-linearity, Attractors and Chaos

A dynamical system can be represented by a state vector, $x = (x_1, \ldots, x_n)$, time t, and an evolution function f that transforms a state at one time to another state at some other time:

[2] The French philosopher and sociologist Auguste Comte introduced positivism in the beginning of the nineteenth century. Positivism asserts that the only authentic knowledge is that which is based on sense experience and positive verification. The German philosopher Edmond Husserl introduced phenomenology in the beginning of the twentieth century as the study of consciousness and conscious experience. Among the most important processes studied by phenomenology are intentionality, intuition, evidence, empathy, and intersubjectivity. The positivism-phenomenology distinction opens the debate on objectivity and subjectivity. Our occidental world based most of our design and engineering on positivism which led to developing a very precise and verifiable syntax, often leaving semantics somewhere behind, perhaps because semantics is full of subjectivity. It is time to re-qualify phenomenology in design and engineering. A few organizations and companies already work in this direction.

$$\frac{dx}{dt} = f(x, t, u, p, q),$$

where $u = (u_1, \ldots, u_m)$ is a control vector, $p = (p_1, \ldots, p_k)$ is a vector of system parameters, and $q = (q_1, \ldots, q_l)$ is a perturbation vector.

Ordinary differential equations are commonly used to model and simulate dynamical systems. Other mathematical approaches include finite state machines, cellular automata, Turing machines, stochastic equations and partial differential equations. When systems can be modeled by these approaches, the temporal behavior of such systems can be analyzed.

There is no universal mathematical tool that enables handling non-linear dynamical systems in their complete form by analogy with, for example, linear systems and linear algebra. It is important to note that Henri Poincaré proposed a number of general ideas and rules that could help scientists choose and implement the corresponding mathematical techniques to solve specific problems in Mathematics. Poincaré put together the qualitative theory of differential equations. Poincaré's findings helped the analysis of non-linear dynamical system using differential equation taking into account their evolutions and critical elements such as limit cycles, attractors, chaos, bifurcations and singularities. I advise the reader to refer to the Santa Fe Institute[3] of complexity, that has already produced excellent material on complexity and non-linear dynamical systems, and other work such as (Mitchell 2008; Holland 1998).

Understanding complexity is difficult. Understanding that a problem may have more than one solution is a first start. Understanding that two slightly different initial conditions may lead to drastically different final results is a second step. Understanding that a small variation in an input may result in huge consequences in non-linear dynamical systems is a third step, and so on. Understanding that the behavior of a complex system cannot be fully predictable can be very negative, but understanding that a complex system has some specific properties is very positive. This is precisely what Poincaré understood (i.e., some complex equations cannot lead to meaningful quantitative solutions, they can lead to informative qualitative solutions or properties). This is why we need to look for interesting properties (or attractors) of complex systems. Lorenz's non-linear dynamical equations of his toy atmosphere led to what he called the butterfly effect, a beautiful attractor (Lorenz 1963).

An **attractor** can be seen as an envelope that "includes" all trajectories of a non-linear dynamical system generated from different initial conditions. Each trajectory, or data set, as related to a specific initial condition is an unpredictable instance. Conversely, the accumulation of several trajectories of a non-linear deterministic system defines an attractor that can be anticipated once it is discovered. Chaos theory is mainly based on these premises. More specifically, chaos theory studies the behavior of dynamic systems that are highly sensitive to initial conditions. An attractor can therefore be also defined as a concentration of chaotic trajectories of a non-linear dynamical system.

[3] The Santa Fe Institute is a private, non-profit, multidisciplinary research and education center, funded in 1994. It is primarily devoted to basic research.

Maturity of a complex system cannot be fully reached until behavioral attractors are identified. Chaos theory provides theoretical models such as strange attractors and the butterfly effect (Poincaré 1890; Lorenz 1963). Systems of systems are embedded into each other, sometimes repeating similar structures at a different level of granularity. This is why Fractal theory can be very useful to better analyze and understand both their structure and dynamics (Mandelbrot 2004). Their multiple interactions manage to develop emergent behaviors that are useful to identify. Finally, emergence cannot be sustained without self-organization (Ashby 1947). Socio-cognitive stability, whether passive or active, is an emergent property of a system of systems. Knowing such properties is crucial because self-organization as a complex system does not require any central authority, as long as organization and coordination rules are well defined and used. Think about a huge flock of birds describing patterns in the sky. Their organization and coordination rules ensure the stability of the whole flock. Reynolds proposed a model for flocking that includes three properties (Reynolds 1987): alignment that consists in moving in the same direction as neighbors; separation and collision avoidance that consists of short-range repulsion; and cohesion that consists of remaining close to each other or long-range attraction.

Natural Versus Artificial Complexity

First, let us make a major distinction between natural and artificial complexity. Natural complexity typically denotes complexity of natural systems such as vegetal and animal living organisms, and geological systems. In contrast, artificial complexity denotes complexity of human-made systems (or artifacts), such as mechanical systems.

Natural complexity is incrementally explored from the outside-in. For example, chemists describe plants using attributes that characterize constant behaviors and processes. They try to discover properties. Exploration is a major process used to uncover natural complexity. Research consists of exploring what the major agents involved in the natural systems being investigated are, whether a living organism or a geological system. Each agent exhibits a behavior that must be identified, interpreted and explained. Agents are usually interconnected among each other and complexity can be explained in terms of interconnectivity. Typically, a natural system cannot be easily decomposed because some connections are crucial for the life and/or integrity of the overall system. However, decomposition and categorization are very important processes commonly used in research. For example, a human body can be decomposed into organs, bones, blood and so on. The main issue is that these components are difficult to study without taking into account the various interconnections between them. This issue is called **separability** (e.g., we cannot study the human heart in isolation since it is necessarily interconnected with other organs in order to work properly on all levels). Although, when the right variables and models are identified, it may become possible to isolate some parts and study them in isolation (e.g., biologists have been trying to learn about cells by culturing

Fig. 5.2 Old clock mechanics. (Stock image courtesy of iStockphoto/Thinkstock)

them in isolation). The closed biological system Biosphere 2,[4] for example, is a 3.14 acre research facility owned by the University of Arizona that is intended to simulate living systems in order to better understand Earth and its place in the universe.

Separability, or non-separability, has already been studied in mathematics, physics, chemistry and physiology. For example, in quantum mechanics, the Schrödinger equation describes how the quantum state of a physical system changes over time. There are some cases in which this equation can be separated into several ordinary differential equations that simplify the resolution of the mathematical system (Eisenhart 1948). In chemistry, a mixture of substances can be transformed into several distinct products through a separation process. Separability needs to be analyzed in conjunction with contextuality (or locality). Claude Bernard, a French physiologist and surgeon, discovered the concept of "milieu intérieur" (eBooks-France, Bernard 2000), which was later coined by Walter Cannon as "homeostasis". A complex system such as the human body consists of a huge number of feedback loops among various components that cannot be partly broken for the health of the whole, and some of them cannot be broken at all, otherwise the whole would die.

Artificial complexity can formally be identified and explored before a system is made. However, we also need to make other distinctions between **hardware complexity** and **software complexity**. Automata have been built for a long time. For example, the clock for example has existed for ages, but for a long time its complexity was only mechanical (hardware complexity).

Mechanical complexity is tractable, decomposable, and linearizable. For example, old-clock mechanics could be decomposed into a set of articulated mechanisms intimately organized (Fig. 5.2). It was not easy to make such clocks work because manual and mental skills were required. However, the complexity resulting from clock design could be tracked (e.g., clock mechanisms could be put on the table and related to each other in a linear way). They could be repaired when they failed. Conversely, modern watches are electronic and software-intensive. Their maintenance does not require any specific skills in mechanics as in the past, since the time system

[4] http://www.b2science.org/about/fact.

Fig. 5.3 Enjoying the complexity of wine! (Stock image courtesy of iStockphoto/Thinkstock)

is a computer program. In contrast, when they fail, the entire software system needs to be changed. Usually they are not repaired, we simply change the watch.

People can transform natural entities to make new varieties of objects. Wine for example is such a generic object. It is made from grapes through fermentation and subtle chemical transformations. The beauty of wine is that it may have various kinds of behavior with respect to the terrain where grapes are cultivated, the evolution of weather during the year and so on. Wine is a complex system, both natural and artificial.

In order to analyze the complexity of a good wine, for example, experts could say something like "a combination of richness, depth, flavor intensity, focus, balance, harmony and finesse" (Fig. 5.3). This pattern can be called the "signature" of the wine being tasted. In chaos theory, we would talk about an attractor. For example, Bordeaux wines, while they can be very different among each other (i.e., they have different "trajectories"), they all are consistent with the same signature (i.e., Bordeaux wines' attractor is very distinguishable). A Bourgogne wine has a different signature for example.

Wine complexity can be described by words such as[5] "acetic, acidic, ageworthy, aggressive, ample, aromatic, astringent, austere, balanced, big, bitter, blockbuster, body, bold, bouquet, buttery, bright, character, clean, complex, concentrated, cooked, corked, crisp, deep, dry, dull, easy-drinking, elegant, fat, flabby, fleshy, focused, fresh, fruity, grassy, green, hard, harsh, heavy, herbal, honeyed, jammy,

[5] http://world-food-and-wine.com/describing-wine.

lean, light, long, madeirized, mature, meaty, mineral, neutral, noble, nose, nutty, oaky, oxidized, petrol, piercing, powerful, racy, rich, ripe, rounded, simple, smooth, soft, sparkling, spicy, steel, stony, structured, subtle, sulphurized, supple, sweet, tannic, tart, toasty, velvety, warm, woody, yeasty... and so on". These words attempt to describe the terminology of wine tasting. Of course, such terminology would become a real ontology if a wine expert used it, someone who knows how to relate the various terms with the interconnected concepts that characterize wine. In particular, experts know that such terms are useful to denote emerging properties resulting from the various relevant transformations and evolutions (of wine in this case). Again, complexity is in the number of independent terms and concepts being used, as well as the interconnections between them.

Today, technology develops very fast and is increasingly interconnected. Not only are artificial systems interconnected, natural and artificial systems are interconnected as well. This is why human-centered design has become tremendously important. Making a good wine requires knowledge, knowhow, experience and risk taking. Wine making can qualify as a **system of systems** (see a workable definition later in this paper). Wine emerges from mechanical and chemical transformations of grapes. These transformations can be considered as sub-systems. In the same way, developing a highly interconnected complex system requires knowledge, knowhow, experience and risk taking. For example, current and future air traffic management systems are systems of systems where there are artificial systems (e.g., onboard and ground systems) and humans (e.g., pilots, air traffic controllers, airworthiness officers, etc.). Such a system of systems evolves where both artificial and natural (sub)-system behaviors are incrementally transformed and adapted to ensure consistency of the whole. It is very difficult to make a good wine because the transformation and adaptation of each agent can be challenging to master. Wine makers need to know how wine agents interact amongst each other. In the same way, in the ATM we need to incrementally understand how ATM agents interact among each other, taking into account that as the number of aircraft increases, the nature of their interconnections evolves.

Toward a Socio-Cognitive Framework

Multi-agent systems have been studied for a long time in artificial intelligence (Ferber et al. 2009; Nair et al. 2003; Ferber 1999; Bradshaw 1997; Minsky 1985). In multi-agent modeling, the first difficulty is to define the various relevant agents, and second their interdependence and relationships, as well as how they interact with each other. Note that in this chapter, I do not want to limit the description of agents to machines or what is now called intelligent (artificial) agents; agents can be natural and artificial. We also need to be able to observe them in the real world in order to acquire their behavior and later on identify their internal mechanisms. They all behave with respect to their intentions (goal-driven behavior and functions) and reactions (event-driven behavior and functions).

Identifying **interaction patterns** among agents and finding out their internal mechanisms is actually an outside–in approach. Internal mechanisms of both artificial and natural agents are algorithms supported by appropriate internal architectures

motivated by observed an/or desirable interactions. Deduced algorithms and architectures of artificial agents are very useful requirements for the design of real systems. Deduced algorithms and architectures of natural agents (typically people) are attributes for determining how they might/should train and work. This is why modeling and simulation of such multi-agent systems are so important very early during the design process of SoSs, to **identify the appropriate interaction space made of appropriate resources and contexts**.

At this point, it is important to make clear that the outside-in approach to making complex systems needs to be supported by a solid framework. The cognitive function framework is one of them (Boy 1998). When a cognitive function is allocated to an agent, it provides a specific role to this agent. If the agent is a postman, his or her main cognitive function (i.e., his or her role), is to deliver the mail. In addition, the postman belongs to the mail services agency (i.e., a system of systems). A cognitive function is also defined within a context (e.g., the postman delivers letters from 8 a.m. to 12 p.m. and 2 p.m. to 5 p.m. every weekday in a well-defined neighborhood). The cognitive function context can then be temporal or spatial, but it also can be normal or abnormal. For example, a strike could be an abnormal context when the postman in duty has longer working hours and a bigger neighborhood. Finally, a cognitive function has resources that support its achievement (e.g., the postman has a bag, a bicycle and a special mental skills to pattern-match addresses on envelopes to street names, house numbers and people's names). The formers are physical resources and the latter (i.e., pattern-matching) is a cognitive resource that is itself a cognitive function. Consequently, a cognitive function can also be a "society of cognitive functions." In some abnormal contexts, such as a strike, postmen on duty cannot do the job by themselves, and need to delegate part of the delivery task to several other people. These people may not be trained and need to be supervised. Postmen on duty need to have cognitive functions such as "training other people to deliver letters," "supervising," "evaluating performance," and so on. Note that the "letter delivery" cognitive function can be decomposed into several other cognitive functions that themselves can be distributed among a set of agents.

The cognitive function framework is structured around two spaces: the context space and the resource space (Boy 2011).

In **context space**, the main entity is a **procedural scenario** or chronological script. A procedural scenario can be described as a tree that may have normal and abnormal branches. The basic entity that is commonly used to develop procedural scenarios is the interaction block or i-Block (Boy 1998). The central component of an i-Block is its context C. The context of an i-Block is a set of (persistent) conditions that need to be satisfied to enable the i-Block. i-Blocks are mutually inclusive (i.e., a context of i-Block is an i-Block). i-Blocks are procedurally organized (i.e., an i-Block is necessarily followed by another i-Block except when it terminates a context). A terminating i-Block has a goal (or an abnormal condition) that is the same as the goal (or an abnormal condition) of its context. Similarly, a starting i-Block has a triggering condition that is the same as the triggering condition of its context.

The triggering condition TC of an i-Block is a set of conditions that activate the i-Block when they are satisfied and the i-Block is enabled. The goal G of an i-Block

is a set of conditions that terminates the activity of an i-Block when they are satisfied. An abnormal condition AC of an i-Block is a set of conditions that terminates the activity of an i-Block when they are satisfied. Therefore, the activity of an i-Block can be terminated either normally when its goal is reached or when an abnormal condition is satisfied.

Therefore, an i-Block can be defined as: iB = {TC, G, AC, C, CF}, where CF is the cognitive function that enables to execute the task assigned to the i-Block). In fact, an iB is the execution of a CF in a given context C. TC, G and AC enable the connection between several iBs.

In the **resource space**, the main entity is the **declarative scenario** or organizational configuration. A declarative scenario can be described by a network of cognitive functions and physical artifacts. The basic entity that is commonly used to develop procedural scenarios is the cognitive function or CF (Boy 1998). A CF can be described by three attributes: its role R, its context of validity C and its resources $\{RES_i\}$. C can be normal C_N or abnormal C_A. A resource can be a cognitive CF or a physical artifact PA. Therefore, a cognitive function can be defined as: $CF = \{R, C, \{CF_j, PA_k\}\}$. In addition, there is a relationship between cognitive functions and agents (i.e., the function allocation issue). Agents have competencies that enable the allocation of appropriate functions whether they are cognitive or physical. We will call this allocation relation Alloc (CF, A), where CF is a cognitive function and A an agent. Therefore, functionally speaking, an agent can be defined by a set of cognitive functions (describing her/his/its competence). Of course, as we already saw in the postman example, cognitive functions can be related to each other (cognitive function network) independently of their allocation to agents. Consequently, there will be a set of constraints that will define possible allocations among agents.

Cognitive functions are not only allocated deliberately, they may also emerge from interactions among agents as necessity. This phenomenon of emergence needs to be captured, analyzed and rationalized (Cognitive Function Analysis, see Boy 1998). Emergent cognitive functions are incrementally added to the cognitive function network together with their allocation to appropriate agents.

Some scenarios, whether procedural, declarative or both, can be generic and can be reused in other scenarios. An HCD architect knows many generic scenarios and is able to detect very quickly what clients want and need. This is why research is needed to develop such generic scenarios in order to use them in human-centered design.

Intrinsic Versus Extrinsic Complexity: The Issues of Maturity

As an example, cars are now equipped with new artificial agents, such as:

- cruise control system that maintains constant speed;
- Anti-lock Braking Systems (ABS), Traction Control Systems (TCS, TRC, ASR), Electronic/Dynamic Stability Control Systems (ESP, ASR, DSC) that keep vehicles on track when surface road conditions are not favorable or help maneuvering with vehicle in off-nominal driving situations and conditions;

- Global Positioning System (GPS) that assists a driver in navigation tasks;
- Line Keeping Assistant (LKA) that enables the car to automatically follow a line on the road;
- collision avoidance system that enables a driver to know how close the car is to other cars surrounding it;
- hand-free telephone kit that enables a driver to communicate with other people outside the car, and so on.

Each system usually supplies a specific function that deals with safety, efficiency and comfort: speed control, navigation assistance, trajectory guidance, collision avoidance, and communication support. Each of these functions are great individually because they are solutions to individual problems, and their individual intrinsic complexity is typically mastered. However in some situations, when these functions are collectively used, they may induce high workload, situation awareness and decision-making issues, as well as action taking problems. Drivers need to share their attention among several artificial agents (i.e., systems) that are not integrated (i.e., drivers have new function allocation tasks to perform in order to manage resulting uncoordinated multi-agent systems). We will say that extrinsic complexity is not mastered. This kind of complexity can be qualified as **human-machine coordination complexity**. Such complexity can be represented and tested using the CF-based socio-cognitive framework presented above.

We now understand that extrinsic complexity as a matter of function allocation among agents. If we see modern car cockpits as systems of systems, there is a need for better understanding how the various systems are interconnected not only at design time but also at use time. In other words, it is not sufficient to verify that systems work well intrinsically, they need to be used safely, efficiently and comfortably. For that matter, it may take some time to understand how the various functions have to be allocated, interconnected and coordinated within the context of driving, and more generally during operations. This logically leads to technology maturity (i.e., technology reliability and robustness) and maturity of practice (i.e., interaction among agents).

Maturity is often difficult to measure because technology constantly changes. The problem is to find out how technology changes; it can be an **evolution** or a **revolution**. Technology evolves when small increments are implemented, and practice changes slightly. A technology revolution is observed whenever jobs that it induces drastically change. When systems (e.g., GPS and LKA) are incrementally introduced in an existing system (e.g., a car), this is usually perceived as an evolution. However, when they become too numerous, the job of the user drastically changes because it is no longer possible to manage these systems without a careful and tedious dynamic function allocation. In addition, the overall situation may become dangerous when the resulting function allocation process is too demanding in real-time. Typically, systems are incrementally added and accumulated until behavior of the overall system of systems drastically changes. At this point, a totally new system emerges and we observe a revolution. New strategies need to be found to handle the emergent system, and eventually redesign it.

What we described above is a matter of SoS **configuration.** However, together with the structure there is function that enables behavior. Complex systems also have behavioral properties such as attraction, fractality, emergence and self-organization. From a complexity science point of view, emergent properties could be seen as behavioral attractors.

Dealing with New Types of Complexity and Uncertainty

Our technological world is always changing. Therefore, people are required to adapt to new types of complexity continuously due to the change of cognitive functions that are statically allocated by the introduction of new technology, and of cognitive functions that dynamically emerge from the use of this technology. A number of complexity-related concepts were elicited from experts in the aviation domain, using appropriate knowledge acquisition methods (Boy 2007). The major concept that emerged was "novelty complexity. "

Complexity and Maturity

Many people interact with a computer everyday without caring about its internal complexity... fortunately! This was not the case barely 30 years ago. Individuals who were using a computer needed to know about its internal complexity from both architectural and software points of view, in order to make it do the simplest things. Computer technology was not as mature as it is today. Computer users had to be programmers. Today, almost everyone can use a computer. However, internal complexity may perhaps become an issue in abnormal or emergency situations.

Internal complexity is about *technology maturity*. Maturity is a very complex matter that deals with the state of evolution of the technology involved, and especially reliability. Are both the finished product and related technology stabilized? Internal complexity of artifacts that are mature and reliable (i.e., available with an extremely low probability of failure), is or not at all perceived. In other words, when you can delegate with confidence and the work performed is successful, the complexity of the delegate is not an issue. However, when it fails, you start to investigate why. You look into the "black-box. " For that matter, the "black-box " should become more understandable (i.e., an appropriate level of complexity must be shown to the user). This implies the user should be able to interpret this complexity. Therefore, either the user is expert enough to handle it or needs to ask for external help. For example, current sophisticated cars and trucks are so computerized that when something is suddenly goes wrong, the driver is typically not able to understand the situation and consequently comes to a decision, perhaps not acting inappropriately. Particular kinds of purposeful information should be provided to avoid even the worst consequences. Various levels of explanations should be available according to context; this is a difficult thing to do.

Perceived complexity is about *practice maturity*. Is this technology adequate for its required use, and how and why do, or don't, users accommodate to and appropriate the technology? Answers to this question contribute to a better understanding of perceived complexity, and further development of appropriate empirical criteria. Perceived complexity is a matter of the relationship between users and technology. Interaction with an artifact is perceived as complex when the user cannot do or has difficulty doing what he or she decides to do with it. Note that users of a new artifact still have to deal with its reliability and availability, which are not only technological, but are also related to tasks, users, organizations and situations. This is why novelty complexity analysis and evaluation require solid structuring into appropriate categories.

A user who interacts with a complex system inevitably builds expertise. He or she cannot interact efficiently with such an artifact as a naive user. There is an adaptation period for new complex tasks and artifacts because complex artifacts such as airplanes are prostheses. The use of such prostheses requires two major kinds of adaptation: mastering capacities that we did not have before using them (e.g., flying or interacting with anyone anywhere anytime); and measuring the possible outcomes of their use, mainly in social terms. All these elements are intertwined, involving responsibility, control and risk/life management. Therefore, provided evaluation criteria cannot be used by just anyone. These criteria must be used by a team of human-centered designers who understand human adaptation. You may ask, who could disagree with this? Today, the straight answer is, the finance-driven decision-makers, just because it is too expensive in the short-term. But, how can we manage maturity with short-term goals?

Maturity Management

It is expected that the complexity of an artifact varies during its life cycle. Therefore, both technology and practice maturities need to be taken into account along the life-cycle axis of a product by all appropriate actors; I call "maturity axis" a sequence of maturity checkpoints and re-design processes up to the entry into service. There are methods that were developed and extensively used to improve the efficiency of software production processes such as Capacity Maturity Model Integration or CMMi (Paulk et al. 1993). CMMi partly contributes to assure technology maturity, in the sense of quality assurance. However, this method does not address directly either internal complexity or perceived complexity of the artifact being developed. This is why maturity checkpoints that involve usability (Nielsen 1993) tests during the whole life cycle are strongly recommended.

"Complexity refers to the internal workings of the system, difficulty to the interface provided to the user—the factors that affect ease of use. The history of technology demonstrates that the way to provide simpler, less difficult usage often requires more sophisticated, more intelligent, and more complex internal workings. Do we need intelligent interfaces? I don't think so: The intelligence should be inside, internal

to the system. The interface is the visible part of the system, where people need stability, predictability and a coherent system image that they can understand and thereby learn" (Norman 2002). Norman's citation is very important today when we have layers and layers of software piled on top of each other, sometimes designed and developed to correct previous flaws of lower layers. Software engineers commonly talk about patches. This transient way of developing artifacts does not show obvious maturity. Technology maturity, and consequently internal complexity, of an artifact can be defined by its integration, reliability, robustness, socio-cognitive stability and availability. As previously stated, it is always crucial to start with good, high-level requirements that, of course, will be refined along the way. Problems arise when those requirements are weak.

Technology-Centered Internal Complexity Versus User-Perceived Complexity: Focusing on Cognitive Stability

Schlindwein and Ray (2004) introduced the distinction between descriptive and perceived complexity as an epistemological problem of complexity. Perceived complexity may block users when they cannot see or anticipate the outcome of their possible actions (i.e., world states are so intertwined that they cannot see a clear and stable path to act correctly). In addition, they may become aware of such complexity after they commit errors and need to recover from them. At this point, it is appropriate to say that real situation awareness is learned in this kind of error-recovery situation.

Therefore, user-perceived complexity is intimately linked to cognitive stability. Technology provides **cognitive stability** when it is either self-recoverable (like the automated spelling checker that automatically correct typos) or supports users with the necessary means to anticipate, interact and recover by themselves. The main problem with self-recoverable systems is reliability and robustness. When they are not reliable or robust enough, people stop using them. We know that users detect most of their errors, and recover from them when they have the appropriate recovery means, whether these means are technological (i.e., tool-based) or conceptual (i.e., training-based). It was observed that in air traffic control, controllers detected 95 % of their errors (Amalberti and Wioland 1997). Obviously, the required level of the user's experience and expertise may vary according to the cognitive stability of the overall human-machine system.

When a person controls a machine, two main questions arise:

1. Are machine states observable (i.e., are the available outputs necessary and sufficient to figure out what the machine does)?
2. Are machine states controllable (i.e., are the available inputs necessary and sufficient to appropriately influence the overall state of the machine)?

A mental model is developed to control a machine, associating observable states to controllable states (Rasmussen 1986; Norman 1986). There is a compromise to be made between controlling a system through a large set of independent observable

states and a small set of integrated observable states. "... The larger the number of degrees of freedom in a system, the more difficult it is to make the system behave as desired (i.e., perceived complexity is higher). Simply counting degrees of freedom, however, oversimplifies the issue. It is the manner in which degrees of freedom interact that determines the difficulty of controlling a system. For example, if the n degrees of freedom of a system are independent of one another, then the controlling system needs only to process an algorithm that is adequate for the control of a single degree of freedom; the algorithm can be replicated n times to control the overall system. Conversely, if the degrees of freedom are dependent (that is, if the effects of specifications of values for a particular degree of freedom depend on the values of other degrees of freedom), then a team of independent controllers is no longer adequate, and more complex control algorithms must be considered. (Jordan and Rosenbaum 1989; Norman 2002).

Cognitive stability is analyzed using the metaphor of stability in physics. Stability can be static or dynamic. Static stability is related to the degrees of freedom (e.g., an object in a three-dimensional world is usually defined by three degrees of freedom). A chair is stable when it has (at least) three legs. Human beings are stable with two legs, but this is a dynamic stability because they have learned to compensate for, often unconsciously, their instability. When an object is disturbed by an external event, there are usually two cases: a case where the object returns to its original position, we say that the object is in a stable state; and a case where the object diverges from its original position, we say that the object is (or was) in an unstable state. When a user acts erroneously, there are two cases: in a case where the user recovers from his or her erroneous action, we say that the user is in a stable state; and in a case where the user does not recover from his or her erroneous action, we say that the user is (or was) in an unstable state.

There are erroneous human actions that may be tolerated, and others that should be blocked. Error tolerance and error resistance systems are usually related to useful redundancy. Error tolerance is always associated with error recovery. There are errors that are acceptable to make because they foster awareness and recovery. However, recovery is often difficult, and sometimes impossible, when appropriate resources are not available. Action reversibility should be put forward and exploited whenever a user can backtrack from an erroneous action, and act correctly. The UNDO function available on most software applications today provides a redundancy to users who detect typos and decide to correct them. Thus, making typos is tolerated, and a recovery resource is available. Error resistance is, or should be, associated to risk. Error-resistance resources are useful in life-critical systems when high risks are possible. They may not be appropriate in low-risk environments because they usually disturb task execution. For example, text processors that provide permanent automatic grammar checking may disturb the main task of generating ideas, and unwanted automatic "corrections" to text that are off the visible screen can go unnoticed. Inappropriate learning and training, poor vigilance, poor feedback, too much interruption, fatigue and high workload are important adverse influences on cognitive stability.

Organizational Complexity: Focusing on Socio-Cognitive Stability

The exploration of organizational automation leads to the investigation of the concept of organizational complexity. The management science literature is certainly rich in definitions of organizational complexity, but they still need to be improved in order to take into account organizational automation. For example, Dooley (2002) defines organizational complexity as "... the amount of differentiation that exists within different elements constituting the organization. This is often operationalized as the number of different professional specializations that exist within the organization. For example, a school would be considered a less complex organization than a hospital, since a hospital requires a large diversity of professional specialties in order to function. Organizational complexity can also be observed via differentiation in structure, authority and locus of control, and attributes of personnel, products, and technologies." In other words, the number and diversity, as well as authority distribution, of agents and, more specifically, cognitive functions involved in the organization are direct contributing factors to organizational complexity.

Luhmann (1995) states "we will call an interconnected collection of elements complex when, because of imminent constraints in the elements' connective capacity, it is no longer possible at any moment to connect every element with every other element... Complexity in this sense means being forced to select; being forced to select means contingency; and contingency means risk". In this definition, connectivity is the problem where channels for communication, cooperation and coordination are crucial. For example, accidents may occur when two different agents use their authority to decide to act in ways that are finally incompatible for the overall product. In fact, these agents were supposed to be independent, but it turned out that their dependency emerged from the situation. Therefore, since emergent cognitive functions are generally situated, they cannot be discovered without operational experience (i.e., either using a simulation facility or in a real-world environment). More specifically, there is an account of two hospital services that were somehow disconnected, and both eventually administered incompatible drugs to the same patient, each of these drugs being totally acceptable independently. Another example can be taken from the aviation domain, where a wiring system was being developed by a division of a large company and was not aware of the integration constraints of another division. In the former case, the lack of connectivity might have caused the death of the patient without the intervention of the patient's family; in the latter case, it caused a drastic delay in the delivery of the product and subsequent financial issues.

The complexity of the product itself has a major influence on the organizational complexity. Designing a large commercial aircraft is not the same as designing an ordinary meal. In order to reduce organizational complexity, a typical strategy of financial management is to divide the overall design and manufacturing work into small pieces that are simple enough to be performed by cheap labor. The main question that remains to be solved is the integration issue that requires both global and specific technology competence. This kind of competence is progressively removed from current industrial organizations and replaced by reporting mechanisms

mainly focused on financial factors. Instead, connectivity in the sense of domain-specific, as well as educated common sense articulation, work should be further developed where people would be involved in a participatory way. Instead of relying implicitly on articulation work performed by a few motivated people, it is crucial to focus the way people communicate, cooperate and coordinate within the organization; this is what I call human-centered continuum in the organization. Therefore, the main issue is to (re)-create a human-centered continuum in the organization instead of dichotomized pieces of a financial puzzle, hoping that these pieces will magically connect among each other in the real life. In fact, socio-cognitive stability is enhanced when people deploy a collaborative involvement toward a mature product.

Socio-cognitive stability (SCS) has been defined as local and global SCS (Boy and Grote 2009). Local SCS is defined as the optimum agent's workload, situation awareness, ability to make appropriate decisions and, finally, correct action execution. It can be supported by appropriate redundancies and various kinds of cognitive support such as trends, relevant situational information and possible actions. Global SCS is defined as the appropriateness of functions allocated to agents, pace of information flows and related coordination. It is very similar to the level of synchronization of rhythms in a symphony. Globally, socio-cognitive support could be found in a safety net that would take into account the evolution of interacting agents and propose a constraining safety envelope in real time.

The major problem is the mismatch between the growing number of interdependencies and the lack of concrete links that should materialize these interdependencies. They are supposed to be glued in the end, but this is where the system does not work because the necessary gluing process is often performed in a hurry, at "the last minute," or the gluing requires expert human judgment, which is excluded from the process. Boy and Grote (2009) proposed a measure of socio-cognitive complexity derived from several contributions such as Latour's account on socio-technical stability (Callon 1991; Latour 1987), emerging cognitive functions (Boy 1998), distributed cognition (Hutchins 1987), and socio-cognitive research and engineering (Hemingway 1999; Sharples et al. 2002). Three kinds of measures were deduced during the PAUSA project for assessing socio-cognitive stability:

1. *time pressure criticality* (e.g., in the air traffic management (ATM) case, the amount of workload that an agent (or a group of agents) requires to stabilize an ATM system after a disturbance) is a workload measure that could be assessed as the ratio between the sum of required times for each action on the total available time (Boy 1983);
2. *complexity* could be characterized by the number of relevant aircraft to be managed per appropriate volumetric zone (AVZ) at each time[6];

[6] An AVZ is calculated with respect to the type of flow pattern (e.g., aircraft crossing, spacing and merging). The definition of such an appropriate volumetric zone requires the assistance of operational ATC controllers. From a socio-cognitive perspective in ATM, complexity should be considered together with capacity. This is what the COCA (COmplexity & CApacity) project

3. *flexibility* could be characterized by the ease of modification of an air-ground contract in real-time—flexibility assessments should guide ATM human-centered automation and organizational setting.

Metrics representing socio-cognitive stability could be added as a fourth kind of measure. Overall, increasing the number of agents and their interdependencies increases complexity and can increase uncertainty, which need to be managed by finding the right balance between reducing uncertainties through centralized planning and coping with uncertainties through decentralized action. It is very difficult and most often impossible, to achieve overall system goals without loose coupling among actors who are required to be both autonomous and intimately coordinated (Grote 2004).

Positivist Versus Phenomenological Approaches

The difficulty in the making of complex systems is mastering novelty and consequently maturity. Product maturity is a matter of high-level requirements and iterative tests during the various design and development processes (Boy 2005). This is why if you start with good architecture in the first place, you are very likely to end up with what you really want. Conversely, if the starting architecture is not well thought out, you will end up with constant modifications to reach satisfaction. The right dose of expertise and common sense is required to define high-level requirements. Afterwards, intrinsic complexity will be mastered if the right engineers are involved at the right time and in the right configuration. In contrast, extrinsic complexity will be a matter of mastering the way various natural and artificial agents will interact among each other. Understanding extrinsic complexity typically leads to redesigning artificial agents (until a stable technology maturity is reached) and the definition of training human agents (until a stable maturity of practice is reached).

During the eighties, Airbus introduced glass cockpits, fly-by-wire and highly automated aircraft technology. Everything was done with a vision, but implemented incrementally using a strong positivist approach (i.e., functions, instruments and systems were added incrementally). This resulted in an accumulation of systems, and more importantly, their limited autonomy transformed the job of a pilot into a management job. Literally, the pilot's job moved from control to management. A new **phenomenon** emerged in the cockpit: systems management. Cognitive engineering was being developed at the same time and rapidly became the scientific support for the analysis of this kind of job evolution. In the making of complex systems it is then important that **systems engineering** (the positivist approach) be carried out taking into account **systems management** (the phenomenological approach).

Systems engineering (SE) focuses on developing processes, which need to be certified (e.g., ISO 9000 and CMMi).[7] SE approaches lead to form filling, database

investigated (Laudeman et al. 1998; Athènes et al. 2002; Masalonis et al. 2003; Hilburn 2004; Cummings and Tsonis 2006).

[7] CMMI (Capacity Maturity Model Integrated) was developed by the Software Engineering Institute at Carnegie Mellon University. This process improvement approach helps integrate traditionally

management, requirements analysis and traceability, model-based system design and performance modeling, planning and project control, software engineering and formal engineering documentation. Very sophisticated tools have been developed to support this kind of approach (e.g., CRADLE),[8] and leads to collaborative system engineering. However, this kind of engineering (positivist) approach cannot be successful without taking into account systems management (i.e., a human-centered design phenomenological approach).

Sometimes people believe that a systemic approach, such as quality assurance, to enterprise management guaranties product maturity at delivery time. This is not true. Reporting requirements could be satisfied while the reported job itself could not be performed correctly. **Syntax has become so important that semantics could play a second role**. In other words, since jobs are dichotomized and related through reporting, the content of the reports is sometimes not verified, the report suffices to be validated (except under pressure when money is at stake). For that matter, it is crucial that agents' expertise be analyzed, anticipated and allocated correctly. Such expertise may actually emerge from practice. Current systemic approaches to system engineering tend to automate the enterprise. It is therefore crucial to make sure that people involved in the use of this human and machine multi-agent automation have the right qualifications and are able to handle their task correctly. In particular, it is important that **organization automation** (Boy and Grote 2011) clearly encapsulates authority sharing (function allocation), worker motivation, and is clearly understood and accepted cooperative work. Goals should be shared and commonly accepted. Based on these premises, very simple tools are typically sufficient to support group activities. People are usually not fully involved in jobs where they do not understand why they need to do things they are asked to do (i.e., where system complexity hides the overall goal and the need for their participation).

In order to re-establish semantics in design and development, it is important to discover the phenomena involved in the systems being developed. Why do Apple products work, and why are they sold all over the world? This is because they are easy to use, they are reliable and usually mature; they create very enjoyable user experiences, they are aesthetically pleasing, and so on. The phenomena behind these tools are typically related to safety, efficiency and comfort. Their intrinsic complexity is certainly very high, but their extrinsic complexity is low (i.e., anybody can use them). More importantly, people do not hesitate to buy them even if they are more expensive. Another example is Google where searching for information is so simple that we now use the verb "google" (e.g., "I will google it to find out more about it"). Google software is certainly very complex (intrinsic complexity), but its use is very simple, efficient and comfortable (extrinsic complexity).

But what caused the success of Airbus, Apple and Google? People! These are people who manage and, to some extent, own these companies. Airbus was created by

---

separate organizational functions, set process improvement goals and priorities, provide guidance for quality processes, and provide a point of reference for appraising current processes (www.sei.cmu.edu/cmmi).

[8] http://www.threesl.com.

a handful of competent and motivated people. They did not own Airbus per se, but they were the real leaders of it. They invented Airbus and made it a success. They worked on a vision (i.e., phenomena such as flight management, and not on short-term money-centered predictions). Apple's co-founder Steve Jobs was a charismatic leader who knew about extrinsic complexity and delegated intrinsic complexity to exceptional experts with a coordinated vision. Google is a new venture lead by real experts in the field of search, and who have exceptional skill in discovering how performance can be improved through motivation and involvement. Google's owners also understood that an information search must be free of charge! These are phenomena. They understood that systems engineering is second to systems management.

If it is important to say that visionary people are crucial, it is also important to investigate the kinds of tools and processes that can support such phenomenological approaches. But before introducing tools and processes, let us describe two useful concepts in more details: affordance and emergence.

It is interesting to observe how new technology leads to a variety of new activities that may often become persistent. Sitting in an airport, I watched three ladies in front of me texting on their cell phones; they never stopped until we were called to embark. One of them was even simultaneously texting on two cell phones. People need to communicate. Since this media **affords** one to do it, it is more than normal that this media is used. In addition, it is boring to wait for a flight in an airport, and texting seems to provide a great opportunity to bridge the gap and ensure a continuity with the normal way of interacting with people while isolated in such circumstances. It was also interesting to notice that even when these ladies put their cell phones in their bags, almost systematically the cell phones called them back, ensuring a continuous interaction.

In addition to the affordance concept, this very quick ethnographical account of cell phone use brings to the forefront another concept, **emergence**. Cell phones afford texting because it is easy to do, cheap and silent (does not disturb neighbors as voice would). People text because it is affordable. In addition, cell phone use is a source of emergent behaviors. Before cell phones, we were not texting! These types of **phenomena** (e.g., texting) emerge from the use of new technology (e.g., cell phones). Whenever such emergent practices become predominant and stable, we generally talk about maturity of practice together with a mature technology. Such phenomena cannot be anticipated (predicted). This is because they emerge from chance and necessity (Monod 1971; Atlan 1970). Emergence of texting as a persistent phenomenon is based on four very simple facts: cell phones are equipped with texting capabilities; cell phones are easy to use for texting; texting is cheap; and texting is silent. We are able to rationalize these three facts afterwards, but did someone think about this combination of facts to make texting a success? I guess nobody did. Texting as a phenomenon emerged from the complexity of our evolving interactive technological society.

It is amazing to observe that modern cell phones have tremendous computing power (i.e., are made of incrementally-engineered software and hardware parts in the positivist sense, and their use induces various kinds of persistent phenomena, texting for example). Another example of emergent phenomena in the use of cell phones is the disappearance of planning in our everyday life. People rely on cell

connectivity instead of planning (e.g., instead of using a shopping list, they call their spouse to ask in real-time what is needed from the grocery store). GPS installed in cars tend to remove the natural ways of searching our way on the road. We rely on GPS because it is a mature instrument that leads to mature practices. However, issues arise when this type of system fails. We can experience such **dependency** when they fail at supporting, guiding and/or mediating us; we can be literally lost.

Human-centered design needs to take into account these phenomena. Since they are emergent by nature, **human-in-the-loop simulation** is a necessary tool and method to discover these phenomena prior to developing the system being designed. For a long time, aeronautics experimental test pilots were the only resources to assess and provide the right modification in aircraft design from this point of view. Today, modeling and simulation tools and methods provide capabilities prior to any product development. The main issue that remains is the use of these tools and methods by potential real end-users. In the same way, I watched the three ladies texting, we should be able to play such scenarios using current modeling and simulation tools and methods.

Modeling and simulation are now inevitable in human-centered design. Until now, ergonomists and human factors specialists were used to evaluating system usability just before delivery. . . and sometimes after to explain incidents and accidents. During the nineties, usability engineering (Nielsen 1993) was extensively developed and commonly used in design. The concepts of horizontal and vertical prototypes were used to test systems globally in breadth and locally in depth. Today, we have modeling and simulation tools that enable us to test usability, in the sense of safety, efficiency and comfort, and during early stages of the design process. For example, the Flacon 7X was fully modeled and simulated before anything was built, and Dassault used the results of such modeling and simulation efforts in the development of the real aircraft. Modeling and simulation are useful to uncover the phenomenological patterns that will actually shape both structure and functions of the system to be developed.

The Difficult Issue of Human-System Integration

We have seen that complexity in complex systems usually comes from an uncontrolled accumulation of systems that may be individually justified but may tremendously increase operational complexity as a whole. This kind of situation typically results from a lack of appropriate **human system integration** (HSI). In this case, HSI is about articulating system management. In highly interconnected systems where people and technology share authority, HSI needs to be considered from the early stages of the design process and throughout the product life cycle. When we talk about a product we also focus on its organizational environment. Again, design is about **technology, organizations and people** (the TOP model).

The outside-in approach promotes the necessity of having **architects** to ensure **integration** from the beginning of the design process. This approach does not disqualify builders who are important assets in the physical human-system integration to

ensure details and robustness of the parts. Architects know how to define integration-driven requirements for builders. Architects typically have a scenario-based approach that integrates technology, organizations and people. They need to master both creativity and rationalization (i.e., they are both artists and engineers). In other words, they need to have a vision inspired by the scenarios and requirements from their potential clients, and technical skills that guarantee concrete construction of their mock-ups. In designing complex systems, we need architects who are curious and able to capture what is really needed from a TOP point of view.

Mathematicians acknowledge abstract objects such as triangles or lines. They manipulate these objects. They combine them to make more complex mathematical objects. They have a notion of "space". A Sobolev space (named after the Russian Mathematician Sergei Sobolev), for example, is a vector space of functions equipped with a norm that measures both the size and smoothness of a function. Partial differential equations are naturally found in Sobolev spaces. Another example is the mathematical concept of a Hilbert space (named after the Russian Mathematician David Hilbert) that generalizes the notion of Euclidian space with any finite or infinite number of dimensions. It can be used to study the harmonics of vibrating strings for instance. This notion of space is very useful to simplify the problem space. In a similar way, human-centered design needs to have this kind of problem resolution space.

Industry traditionally used the army type model as a problem solving space. A general is at the top of a cascade of officers down to soldiers. In this kind of space, no transversal communication is theoretically possible. Information flows are generally downstream, with a few information flows going upstream. Industry has evolved for many reasons that include service-oriented production, management of objectives and over-specialization of work. Instead of soldiers, we now have specialized musicians. The Orchestra Model is progressively replacing the army model (Boy 1991, 2009); this evolution will be more explained in Chap. 6. Let us describe the Orchestra Model as a problem solving space. In order to play a symphony, musicians need to understand the same music theory, read scores that are written by a composer, be coordinated by a conductor, and be proficient in one instrument. These four norms are mandatory. Using this analogy, we can rapidly envision the trouble of some of our contemporary companies. Do they have a music theory? Do they have composers? Do they have conductors? Are the musicians trained to play with others? These are many questions that require clear answers, taking into account that the old army model often overlaps the new orchestra model.

Three Examples of Complex Systems

This chapter cannot be terminated without taking three examples of complex systems that our twenty-first century currently develops: genetically modified organisms (GMOs), terrorism and finance-driven corporations. How did we end up making such complex systems?

GMOs are products of science, a combination of biology and computer science. We can make plants that resist weather (e.g., protect plants from frost damage, and other kinds of external aggressions by altering their DNA molecules). New genes can be created. Natural things are now artificial! We ended up creating complex systems that improve plant protection, and agriculture benefits.

If this kind of technology is of interest because agricultural production can be controlled more easily, economical control of such production is now strongly dependent on highly profitable companies that have the power to directly control local agriculture worldwide. When a country accepts this technology it becomes dependent on it for a long time (extrinsic complexity). We do not know the impact of such products on human beings medically speaking (intrinsic complexity). Product maturity is probably not yet reached, and science needs to tell us. For users, maturity of practice requires more attention since some countries have accepted GMOs and others are still reluctant.

The twentieth century was the century of two conventional world wars, plus a cold war, with clear divisions between the war makers. War makers were well identified with clear ideologies and objectives. Since 9–11, norms have changed. We passed from a war space to another one. War makers are more difficult to identify. They are likely to act on civil grounds with no anticipation. Uncertainty is more extreme compared to years past. In addition, we sometimes do not know if it is military or civilian activities. Complexity comes from ignorance, and opponents count on it. Complexity is also generated by solutions that the occidental world has set up to resist terrorism. Terrorist plans are almost always new. They act independently from each other. It is difficult to find a central organization behind actual actors.

We sometimes tend to design defenses that are obsolete by the time of use. The Maginot Line was built after World War I to protect France from possible German invasion. This defense assumed that war would be static and defensive like World War I. Unfortunately, the Germans invaded Belgium and consequently went around the Maginot Line, which became totally ineffective for this type of dynamic and offensive war.

Terrorism is not new. So why has it become a modern fact? A new type of complexity emerges from the way information is conveyed by media. Everything is going faster, and fear is sometimes exploited to make information more “exciting”. In addition, security control in airports has become a giant business. Airfare is more expensive, travel is more annoying, and security is not guaranteed. Complexity has tremendously increased in order to insure security, without satisfactory results. Why? It is the same problem as the fundamental problem of safety that we have treated in aviation. We are using technology and organization that we know, avoiding to investigate the nature of phenomena behind the symptoms. Why terrorism? Many answers can be given to this question. I strongly believe that people need tangible spiritual models that give them a reason to live. When this is not the case, people are tempted to join groups that show strong beliefs, whether they are ideologies or religions, for example. It is therefore easy to train them and make them terrorists. The solution is not in the short-term as most of us believe. It is in the longer term by re-creating harmony, empathy and cooperation among people. What is the new music

theory that we need to set up? Who are the composers? Who are the conductors? What are the profiles of these new musicians?

Finance-driven corporations have progressively emerged from the evolution of the worldwide market economy. For the last 20 years, management in large companies drastically changed from technology-driven to finance-driven leadership. Whether it was in aeronautics, the car industry or other technology makers, leaders were almost always technicians, or former technicians who evolved into management. Business managers have taken the lead, and more importantly finance people run the show. Shareholders have the real leadership of corporations today, and systems engineering provides tools and techniques that literally automate organizations to achieve shareholders' goals. However, this automation is performed using almost only financial variables; technical variables have become peripheral and human requirements are almost not considered. Excel spreadsheets now constitute the mediating representation and tool through which employees interact with the overall system. It results in impersonal interaction among the various agents of the entire enterprise. Motivation tremendously decreased because work has been dichotomized to the extreme in order to enable appropriate top-down financial management. Music theory is only finance-based. Accountability is at the financial level and not at the technical level any longer. Reporting has also become more important than the work it reports. There is lack of technical composers and conductors, and musicians require motivation and leadership. Analyzing this evolution of our corporations worldwide as complex systems, we immediately observe that they very often have inside-out approaches to system design and development by lack of top-level socio-technical leadership. They are definitely not human-centered, nor they are developing human-centered products. The main issue is that people are "still there" to do some kind of work and use generated products.

It is time to react to this evolution and create frameworks that make human-centered design possible. We need to replace short-term prediction by **vision**! Short-term prediction works well with numbers locally, but does not work at all with our lives globally and in the long-term. We need visionary people who set goals (composers) and create the mandatory motivation that enables the construction of a better world led by leaders who respect people and nature (conductors). Human beings have always created and developed artifacts. It is now very important to focus on the complexity of our combined artificial and natural world.

References

Amalberti, R., & Wioland L. (1997). Human error in aviation. Invited speech, International aviation safety conference (Iasc-97) Rotterdam Airport, The Netherlands. In H. Soekkha (Ed.), *Aviation Safety* (pp. 91–108). Utrech: Vsp.

ASA (1994). *Au temps de Clément Ader*, ouvrage coordonné par l'Académie de l'Air et de l'Espace. ISBN 2-87717-044-6.

Ashby, W. R. (1947). Principles of the self-organizing dynamic system. *Journal of General Psychology*, *37,* 125–128.

Athènes, S., Averty, P., Puechmorel, S., Delahaye, D., & Collet, C. (2002). ATC complexity and controller workload: Trying to bridge the gap. In J. Hansman, S. Chatty, & G. Boy (Eds.), *Proceedings of HCI-Aero'02*, Boston.

Atlan, H. (1970). *L'organisation biologique et la théorie de l'information*. Paris: Herman.

Bernard, C. (2000). Principe de médecine expérimentale (in French). http://www.laphilosophie.fr/ebook/Bernard,%20Claude%20-%20Principes%20de%20m%E9decine%20experimentale.pdf. Accessed 11 Jan 2012.

Boy, G. A. (1983). The MESSAGE system: A first step toward computer-supported analysis of human-machine interactions (in French). *Le Travail Humain Jounal, 46*(2).

Boy, G. A. (1991). Advanced interaction media as a component of everyday life for the Coming Generation. *Proceedings of the World Marketing Congress*. Tokyo: Japan Management Association.

Boy, G. A. (1998). *Cognitive function analysis*. USA: Greenwood/Ablex. ISBN 9781567503777.

Boy, G. A. (2005). Knowledge management for product maturity. *Proceedings of the International Conference on Knowledge Capture* (K-Cap'05). Banff, Canada. October. New York: Also in ACM Press Digital Library, (http://dl.acm.org).

Boy, G. A. (2009). The Orchestra: A conceptual model for function allocation and scenario-based engineering in multi-agent safety-critical systems. *Proceedings of the European Conference on Cognitive Ergonomics*. Finland: Otaniemi, Helsinki area, (30 September-2 October).

Boy, G. A. (2011). Cognitive function analysis in the design of human and machine multi-agent systems. In G. A. Boy (Ed.), *Handbook of human-machine interaction: A human-centered design approach*. Aldershot: Ashgate.

Boy, G. A., & Grote, G. (2009). Authority in increasingly complex human and machine collaborative systems: Application to the future air traffic management construction. *In the Proceedings of the 2009 International Ergonomics Association World Congress*, Beijing.

Bradshaw, J. (Ed.). (1997). *Software agents*. Cambridge: MIT.

Callon, M. (1991). Techno-economic networks and irreversibility. In J. Law (Eds.), *A sociology of monsters: Essays on power, technology and domination* (pp. 132–161) London: Routledge.

Cummings, M. L., & Tsonis, C. G. (2006). Partitioning complexity in air traffic management tasks. *International Journal of Aviation Psychology, 16*(3), 277–295.

Dooley, K. (2002). Organizational complexity. In M. Warner (Ed.), *International encyclopedia of business and management* (pp. 5013–5022) London: Thompson Learning.

Eisenhart, L. P. (1948). Enumeration of potentials for which one-particle Schrodinger equations are separable. *Physics Review, 74,* 87–89.

Ferber, J. (1999). *Multi-agent system: An introduction to distributed artificial intelligence*. Harlow: Addison Wesley Longman. ISBN 0-201-36048-9.

Ferber, J., Stratulat, T., & Tranier, J. (2009). Towards an integral approach of organizations in multi-agent systems: the MASQ approach. In V. Dignum (Ed.), *Multi-agent systems: Semantics and dynamics of organizational models*. IGI.

Ford, K. M., Glymour, C., & Hayes, P. J. (1997). Cognitive prostheses. *AI Magazine* (Vol. 18 Issue 3). Fall.

Grote, G. (2004). Uncertainty management at the core of system design. *Annual Reviews in Control, 28*(2), 267–274.

Hemingway, C. J. (1999). Toward a socio-cognitive theory of information systems: An analysis of key philosophical and conceptual issues, *IFIP WG 8.2 and 8.6 Joint Working Conference on Information Systems: Current Issues and Future Changes*. Finland: IFIP, pp. 275–286.

Hilburn, B. (2004). Cognitive complexity in air traffic control: A literature review. Project COCA—COmplexity and CApacity. EEC Note No. 04/04.

Holland, J. H. (1998). *Emergence: From chaos to order*. Reading: Perseus Books.

Hutchins, E. (1995). How a cockpit remembers its speeds. *Cognitive Science, 19,* 265–288.

Jamshidi, M. (2005). System-of-systems engineering—A definition. *IEEE SMC*, 10–12. http://ieeesmc2005.unm.edu/SoSE_Defn.htm. Accessed 20 Feb 2012.

Jordan, M. I., & Rosenbaum, D. A. (1989). Action. In M. I. Posner (Ed.), *Foundations of cognitive science*. Cambridge: The MIT Press.

Latour, B., (1987). S*cience in action: How to follow scientists and engineers through society*. Cambridge: Harvard University Press.

Laudeman, I. V., Shelden, S. G., Branstrom, R., & Brasil, C. L. (1998). *Dynamic density. An air traffic management metric*. California: National Aeronautics and Space Administration, Ames Research Center, NASA/TM-1998-112226.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Science, 20*(2), 130–141.

Luhmann, N. (1995) *Social systems* (trans: J. Bednarz & D. Baecker). Stanford: Stanford Press.

Luzeau, D., & Ruault, J. R. (Eds.). (2008). *Systems of systems*. Hoboken: Wiley. ISBN 978-1-84821-164-3.

Mandelbrot, B. (2004). *Fractals and chaos*. Berlin: Springer. ISBN 9780387201580.

Masalonis, A. J., Callaham, M. B., & Wanke, C. R. (2003). Dynamic density and complexity metrics for realtime traffic flow management. Presented at the ATM 2003 Conference, 5th EUROCONTROL/FAA ATM R&D Seminar, Budapest.

Minsky, M. (1985). *The society of mind*. New York: Simon and Schuster.

Mitchell, M. (2008). *Complexity: A guided tour*. NewYork: Oxford University Press.

Mitchell, M. (2009). *Complexity: A guided tour*. Oxford: Oxford University Press. ISBN 0195124413.

Monod, J. (1971). *Chance and necessity: An essay on the natural philosophy of modern biology* (trans: A. Wainhouse). Alfred A. Knopf (originally published as *Le hasard et la nécessité.* Paris: Le Seuil, 1970).

Nair, R., Tambe, M., & Marsella, S. (2003). Role allocation and reallocation in multiagent teams: Towards a practical analysis. AAMAS'03, July 14–18, 2003, Melbourne, Australia.

Nielsen, J. (1993). *Usability engineering*. Boston: Academic Press.

Norman, D. A. (1986). Cognitive engineering. In D. Norman, S. Draper, (Eds.), *User-centered system design* (pp. 31–61). Hillsdale: Lawrence Erlbaum Associate.

Norman, D. A. (2002). Complexity versus difficulty: Where should the intelligence be?, in *IUI'02 International Conference on Intelligent User Interfaces*. Miami.

Paulk, M., Curtis, B., Chrissis, M., & Weber, C. (1993). *Capability maturity model for software* (*Version 1.1*). Technical Report CMU/SEI-93-TR-024.

Prigogine, I. (1997). *The End of Certainty*. The Free Press, New York.

Poincaré, J. H. (1890). Sur le problème des trois corps et les équations de la dynamique. Divergence des séries de M. Lindstedt. *Acta Mathematica, 13,* 1–270.

Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction—An Approach to Cognitive Engineering*. Amsterdam: North Holland.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *Computer Graphics, 21*(4), 25–34.

Schlindwein, S. L., & Ray, I. (2004). Human knowing and perceived complexity: Implications for systems practice. *E:CO, 6,* 27–32.

Sharples, M., Jeffery, N., du Boulay, J. B. H., Teather, D., Teather, B., & du Boulay, G. H. (2002). Socio-cognitive engineering: A methodology for the design of human-centered technology. *European Journal of Operational Research, 136*(2), 310–323.