30

Developing and Managing Complex, Evolving Information Infrastructures

30.1 Introduction

The Growth of Complexity in ICT Solutions, the Emergence of Information Infrastructures, and the Need for a New Paradigm

30.2 Socio-Technical Complexity and Theories of Complexity

30.3 Design Challenges and Management “Tools”

30.4 Three Examples of Evolving IIs

30.5 Conclusion

References

Ole Hanseth
University of Oslo

30.1 Introduction

30.1.1 The Growth of Complexity in ICT Solutions, the Emergence of Information Infrastructures, and the Need for a New Paradigm

Increased processing power and higher transmission and storage capacity have made it possible to build increasingly integrated and versatile information technology (IT) solutions with dramatically increased complexity (BCS/RAE 2004, Hanseth and Ciborra 2007, Kallinikos 2007). The complexity of IT solutions has also been continuously growing as existing systems, new and old, have been increasingly integrated with each other. Complexity can be defined here as the dramatic increase in the number and heterogeneity of included components, relations, and their dynamic and unexpected interactions in IT solutions (c.f. Hanseth and Lyttinen 2010).

The Software Engineering Institute (SEI) at Carnegie-Mellon University describes this trend as the emergence of ultra-large-scale (ULS) systems (Northrop 2006). The report argues that these ULS systems will push far beyond the size of today’s systems and systems of systems by every measure:

- Number of technological components of various kinds
- Number of people and organizations employing the system for different purposes
• Number of people and organizations involved in the development, maintenance, and operations of the systems
• Amount of data stored, accessed, manipulated, and refined
• Number of connections and interdependencies among the elements involved

The report argues further that the sheer scale of ULS systems will change everything: ULS systems will necessarily be decentralized in a variety of ways, developed and used by a wide variety of stakeholders with conflicting needs that it will be evolving continuously, and constructed from heterogeneous parts. Further, people will not just be users of a ULS system; they will be elements of the system. The acquisition of a ULS system will be simultaneous with its operation and will require new methods for control. These characteristics are, according to the report, emerging in today’s systems of systems; in the near future, they will dominate.

The SEI report states that the scale of ULS systems presents challenges that are unlikely to be addressed adequately by incremental research within the established paradigm. Rather, they require a broad new conception of both the nature of such systems and new ideas for how to develop them. We will need to look at them differently, not just as systems or systems of systems, but as socio-technical ecosystems.

The growth in complexity has brought to researchers’ attention novel mechanisms to cope with complexity, such as architectures and modularity and standards (Parnas 1972, Schmidt and Werle 1998, Baldwin and Clark 2000). Another, more recent, line of research has adopted a more holistic, socio-technical and evolutionary approach, putting the growth in the combined social and technical complexity at the centre of an empirical scrutiny (see, e.g., Edwards et al. 2007). These scholars view these complex systems as new types of IT artifacts and denote them with a generic label of information infrastructures (IIs) (e.g., Star and Ruhleder 1996, Hanseth and Lytinen 2010, Tilson et al. 2010).

Hanseth and Lytinen (2010) define an II, consistent with the aforementioned characterization of ULS, as a shared, open (and unbounded), heterogeneous, and evolving socio-technical system (called installed base) consisting of a set of IT capabilities and their users, operations, and design communities. Typical examples of IIs are the Internet, solutions supporting the interaction among manufacturers along a supply chain, and portfolios of integrated applications in organizations.

### 30.1.2 Key Characteristics of IIs

The most distinctive feature of IIs and ULSs is their overall complexity. Managing IIs, then, is about understanding and managing complexity. More specifically: IIs, like all complex systems, are evolving and not designed from scratch. So managing IIs means managing their evolution.

IIs are radically different from how information and software systems are presented in the literature. Infrastructures have no life cycle—they are “always already present.” This is strictly true of some infrastructures, such as our road infrastructure that has evolved through modifications and extensions ever since animals created the first paths. IT infrastructures certainly have a much shorter history, but an II like the Internet has now been around and constantly evolving for roughly 40 years. The same is the case for IIs like portfolios of integrated information system (IS) (often numbering in the thousands) in larger organizations. Developing IIs, then, requires approaches that are different from the traditional “design from scratch” ones (Edwards et al. 2007, Hanseth and Lytinen 2010, Tilson et al. 2010). It requires somewhat opposite approaches—it is about modifying (changing and extending) the installed base, existing IIs, so that the installed base evolves as far as possible toward what is desired (user requirements).

The evolution of IIs involves regularly a large number of actors. All these actors cannot be strictly controlled from one single point (like, for instance, a manager at the top of a hierarchically structured project organization). They will often act independently. In the case of the Internet, there are thousands, if not millions, of actors developing new services of the top of the Internet or adding new features to lower level services, like, for example, quality of service mechanisms. This means that even though there
are many institutions (like ICANN, IETF, etc.) that are involved in the governance of the Internet, the Internet is not evolving in strictly planned or controlled way. Its evolution is mostly the aggregated result of the various autonomous actors’ actions. Institutions having responsibility for the governance of the Internet can have impact and shape its evolution in a way similar to how we may influence the growth of an organism or a piece of land. The same is the case for large application portfolios. The individual applications and their relations (degree of integration) will change continuously, and no single actor will have a total overview of these changes. Accordingly, we call our approach to how we can shape the evolution of an II “installed base cultivation” (Hanseth and Lyytinen 2010).

If infrastructures are “always already present,” are new infrastructures never emerging? Was not the Internet a new infrastructure at some point in time? Yes, of course. New IIs are indeed emerging. It happens primarily in two ways. One, a system may be growing in terms of number of users and along that path gradually changing from being a system (or application) of limited reach and range into a large scale II. E-mail, for instance, was introduced into the Internet (or rather the Arpanet) at a time when the net consisted of only four computers. Those computers (and the e-mail service) did not constitute an II, but a distributed system of limited complexity. But as the net was growing in terms of computers connected, services, developers, and users, it was taking on more and more the character of an infrastructure. The other way an II may emerge has been seen in most large organizations. Over time, the number of applications has continuously been growing at the same time as old and new applications have been increasingly more integrated. During this process, the application portfolio is growing from the first stand-alone one to a few loosely integrated ones and toward an increasingly more complex II.

The next section will present theories of complexity that help us understand the dynamics of IIs followed by a section giving a brief outline of what I see as the key challenges related to the management of the evolution of IIs and what kind of “tools” that are available for this task. Finally, I will present three examples of II evolution and how the choices of strategy and management tools have shaped the evolution of these IIs.

### 30.2 Socio-Technical Complexity and Theories of Complexity

In this section, I will present three theories of complexity that help us understand the core issues related to IIs.

#### 30.2.1 Actor Network Theory

Actor network theory (ANT) was originally developed as a “toolbox” for studying processes from individual experiments to a situation where scientific facts or theories are universally accepted as such. This “toolbox” consists of a set of concepts for describing how relations between elements of various kinds are established in a process where heterogeneous networks are constructed. Central concepts in early ANT research are closure, enrolment, and alignment. Specifically, closure indicates a state where consensus emerges around a particular technology. Closure stabilizes the technology by accumulating resistance against change. It is achieved through a negotiation process and by enrolling actors/elements of various kinds into a network and translating (re-interpreting or changing) them so that the elements are aligned, that is, fit together, in a way that supports the designers’ intentions.

The early ANT studies can be said to have focused on complexity in the sense that they spelled out the rich and complex relations between the scientific and the technological on the one hand, and the social on the other, related to the making of scientific theories and technological solutions. ANT has been used, for instance, in research on the negotiation of IS standards and the embedding of their local context of development and use (Star and Ruhleder 1996, Hanseth and Monteiro 1997, Timmermans and Berg 1997, Bowker and Star 1999, Fomin et al. 2003).

Since their emergence in the early 1980s, ANT and ANT research have evolved beyond their (so-called) “managerial” approach, which focuses on how a single actor-network is aligned by a dominating central
actor (Law 2003b). Complexity has been addressed more explicitly as the focus has turned to the dynamics unfolding when independent actors try to align different but intersected actor-networks (Latour 1988, Star and Griesemer 1989, Law and Mol 2004, Law and Urry 2004, Law 2003a). This has happened as attention has moved toward more complex cases where order and universality cannot be achieved in the classical way.* These cases are described as "worlds," which are too complex to be closed and ordered according to one single mode or logic. There will only be partial orders, which are interacting in different ways, or interconnected and overlapping subworlds, which are ordered according to different logics. The interconnectedness of the subworlds means that when one is trying to make order in one subworld by imposing a specific logic, the same logic is making disorder in another—an order also has its disorder (Berg and Timmermans 2000, Law 2003b). Rather than alignment, stabilization, and closure, the keywords are now multiplicities, inconsistencies, ambivalence, ambiguities, and fluids (Law and Mol 2002, Law 2003a). Mastering this new world is not about achieving stabilization and closure, but rather about more ad hoc practices—"ontological choreography" of an ontological patchwork (Cussins 1998). This approach has been applied to studies of cases such as train accidents (Law 2003a), a broad range of high-tech medical practices (Mol and Berg 1998), and interdisciplinary research (Star and Griesemer 1989). This approach to complexity has also been applied to analyzing the challenges, not to say impossibility, of achieving closure and stabilization in relation to IIs and ICT standards (Aanestad and Hanseth 2000, Alphonse and Hanseth 2010).

30.2.2 Complexity Science

Just like the ANT community, the emergence of order has been in focus within the field of complexity science. However, while the first has focused on order making where humans are actively involved, the latter has had a focus on how order emerges without any "intelligent designer" involved like, for instance, the order found in complex organisms and "animal societies" like bee hives and ant heaps. But important contributions to complexity science have been made from studies of more social phenomena, in particular within the economy, like financial markets and also, an issue central to this book, standardization (David 1986, Arthur 1994). Complexity science is made up of a broad range of disciplines such as chaos theory and complex adaptive systems (CAS). CAS is concerned with the dynamic with which complex systems evolve through adaptation and is increasingly used in organizational studies, for example, in health care. CAS is made up of semiautonomous agents with the inherent ability to change and adapt to other agents and to the environment (Holland 1995). Agents can be grouped, or aggregated into meta-agents, and these can be part of a hierarchical arrangement of levels of agents. Agents can respond to stimuli—they behave according to a set of rules (schema).

Adaptation is the process whereby the agent fits into the environment and the agent as well as the CAS undergoes change. Adaptation—and creativity and innovation—is seen as being optimal at "the edge of chaos" (Stacey 1996), or more generally, adaptation occurs within the zone of complexity, which is located between the zone of stasis and the zone of chaos (Eoyang 1996). Dooley (1996) suggests that CAS behave according to three principles: order is emergent, the system's history is irreversible, and the system's future is unpredictable.

Overall, complexity science investigates systems that adapt and evolve as they self-organize through time (Urry 2003). Central to the emergence of orders are attractors, that is, a limited range of possible states within which the system stabilizes. The simplest attractor is a single point. There are also attractors with specific shapes, which are called "strange attractors," that is, unstable spaces to which the trajectory of dynamical systems are attracted through millions of iterations" (Capra 1996).

* John Law and Annamarie Mol (2002, p. 1) define complexity as follows: "There is complexity if things relate but don’t add up, if events occur but not within the process of linear time, and if phenomena share a space but cannot be mapped in terms of a single set of three-dimensional coordinates." This definition is very brief and rather abstract, but is in perfect harmony with Cillier's definition presented earlier.
Orders emerge around attractors through various feedback mechanisms, including increasing returns, and through path-dependent processes of many small steps that may end in lock-in situations (David 1986). Some steps may be crucial in the sense that they may force the process in radically different (unexpected) directions. Such points are called tipping or bifurcation points (Akerlof 1970). The existence of such points makes the evolution of complex systems nonlinear in the sense that small changes in a system at one point in time may lead to hugely different consequences at a later point in time. Many of these concepts have been presented to the IS community by Shapiro and Varian (1999) under the label information economy. They will be explained in the subsequent sections.

30.2.2.1 Increasing Returns and Positive Feedback

Increasing returns mean that the more a particular product is produced, sold, or used, the more valuable or profitable it becomes. Infrastructure standards are paradigm examples of products having this characteristic. The development and diffusion of infrastructural technologies are determined by “the overriding importance of standards and the installed base compared to conventional strategies concentrating on programme quality and other promotional efforts” (Grindley 1995: 7).

A communication standard’s value is to a large extent determined by the number of users using it—that is, the number of users you can communicate with if you adopt the standard. This phenomenon is illustrated by well-known examples such as Microsoft Windows and the rapid diffusion of the Internet. Earlier examples are the sustainability of FORTRAN and COBOL far beyond the time when they had become technologically outdated.

The basic mechanism is that the large installed base attracts complementary products and makes the standard cumulatively more attractive. A larger base with more complementary products also increases the credibility of the standard. Together these make a standard more attractive to new users. This brings in more adoptions, which further increases the size of the installed base, and so on (Grindley 1995: 27).

Further, there is a strong connection between increasing-returns mechanisms and learning processes. Increased production brings additional benefits: producing more units means gaining more experience in the manufacturing process, achieving greater understanding of how to produce additional products even more cheaply. Moreover, experience gained with one product or technology can make it easier to produce new products incorporating similar or related technologies. Accordingly, Shapiro and Varian (1999) see positive feedback as the central element in the information economy, defining information as anything that may be digitized. An information good involves high fixed costs but low marginal costs. The cost of producing the first copy of an information good may be substantial, but the cost of producing (or reproducing) additional copies is negligible. They argue that the key concept in the network economy is positive feedback.

30.2.2.2 Network Externalities

Whether real or virtual, networks have a fundamental economic characteristic: The value of connecting to a network depends on the number of other people already connected to it. This fundamental value proposition goes under many names: Network effects, network externalities, and demand-side economies of scale. They all refer to essentially the same point: Other things being equal, it is better to be connected to a bigger network than a smaller one (Shapiro and Varian 1999).

Externalities arise when one market participant affects others without compensation being paid. In general, network externalities may cause negative as well as positive effects. The classic example of negative externalities is pollution: My sewage ruins your swimming or drinking water. Positive externalities give rise to positive feedback (Shapiro and Varian 1999).

30.2.2.3 Path Dependency

Network externalities and positive feedback give rise to a number of more specific effects. One such is path dependence. Path dependence means that past events will have large impacts on future development, and events that appear to be irrelevant may turn out to have tremendous effects (David 1986).
For instance, a standard that builds up an installed base ahead of its competitors becomes cumulatively more attractive, making the choice of standards “path dependent” and highly influenced by a small advantage gained in the early stages (Grindley 1995: 2). The classical and widely known example illustrating this phenomenon is the development and evolution of keyboard layouts, leading to the development and \textit{de facto} standardization of QWERTY (David 1986).

We can distinguish between two forms of path dependence:

1. Early advantage in terms of numbers of users leads to victory
2. Early decisions concerning the design of the technology will influence future design decisions

The first one has already been mentioned earlier. When two technologies of a kind where standards are important—such as communication protocols or operating systems—are competing, the one getting an early lead in terms of number of users becomes more valuable for the users. This may attract more users to this technology, and it may win the competition and become a \textit{de facto} standard. The establishment of Microsoft Windows as the standard operating system for PCs followed this pattern. The same pattern was also followed by the Internet protocols during the period they were competing with OSI protocols.

The second form of path dependence concerns the technical design of a technology. When, for instance, a technology is established as a standard, new versions of the technology must be designed in a way that is compatible (in one way or another) with the existing installed base. This implies that design decisions made early in the history of a technology will often live with the technology as long as it exists. Typical examples of this are various technologies struggling with the backward compatibility problem. Well-known examples in this respect are the different generations of Intel’s microprocessors, where all later versions are compatible with the 8086 processor, which was introduced into the market around 1982.

Early decisions about the design of the Internet technology, for instance, have had a considerable impact on the design of new solutions both to improve existing services and to add new ones to the Internet. For example, the design of the TCP/IP protocol constrains how improved solutions concerning real-time multimedia transfer can be designed and how security and accounting services can be added to the current Internet.

\section*{30.2.2.4 Lock-In: Switching Costs and Coordination Problems}

Increasing return may lead to yet another effect: lock-in. Lock-in means that after a technology has been adopted, it will be very hard or impossible to develop competing technologies. “Once random economic events select a particular path, the choice becomes locked-in regardless of the advantages of alternatives” (Arthur 1994). In general, lock-in arises whenever users invest in multiple complementary and durable assets specific to a particular technology. We can identify different types of lock-in: contractual commitments, durable purchases, brand-specific training, information and databases, specialized suppliers, search costs, and loyalty programmes (Shapiro and Varian 1999). We can also say that lock-ins are caused by the huge switching costs or by coordination problems (or a combination of these) that would be incurred when switching from one standardized technology to another.

Switching costs and lock-ins are ubiquitous in ISs, and managing these costs is very tricky both for sellers and buyers. For most of the history of computers, customers have been in a position where they could not avoid buying (more or less) all their equipment and software from the same vendor. The switching costs of changing computer systems could have been astronomical—and certainly so high that no organization did. To change from one manufacturer (standard) to another would imply changing all equipment and applications at the same time. This would be very expensive—far beyond what anybody could afford. But it would also be an enormous waste of resources, because the investments made have differing economic lifetimes, so there is no easy time to start using a new, incompatible system. As a result, others face switching costs, which effectively lock them into their current system or brand (Shapiro and Varian 1999).
Developing and Managing Complex, Evolving Information Infrastructures

Switching costs also go beyond the amount of money an organization has to pay to acquire a new technology and install it. Since many software systems are mission critical, the risks in using a new vendor, especially an unproven one, are substantial. Switching costs for customers include the risk of a severe disruption in operations.

Lock-in is not only created by hardware and software. Information itself—its structures in databases as well as the semantics of the individual data elements—is linked together into huge and complex networks that create lock-ins. One of the distinct features of information-based lock-in is that it proves to be so durable: equipment wears out, reducing switching costs, but specialized databases live on and grow, increasing lock-in over time (Shapiro and Varian 1999).

The examples of lock-ins and switching costs mentioned so far are all related to infrastructures that are seen as local to one organization. As infrastructures and standards are shared across organizations, lock-in problems become even more challenging.

Network externalities make it virtually impossible for a small network to thrive. But every network has to start from scratch. The challenge to companies introducing new but incompatible technology into the market is to build a network size that overcomes the collective switching costs—that is, the combined switching costs of all users. In many information industries, collective switching costs are the biggest single force working in favor of incumbents. Worse yet for would-be entrants and innovators, switching costs work in a nonlinear way: Convincing 10 people connected in a network to switch to your technology is more than 10 times as hard as getting one customer to switch. But you need all 10, or most of them: No one will want to be the first to give up the network externalities and risk being stranded. Precisely because various users find it so difficult to coordinate a switch to an incompatible technology, control over a large installed base of users can be the greatest asset you can have.

But lock-in is more than cost. As the community using the same technology or standard grows, switching to a new technology or standard becomes an increasingly larger coordination challenge. The lock-in represented by QWERTY, for instance, is most of all a coordination issue. It is shown that the individual costs of switching are marginal (David 1986), but, as long as we expect others to stick to the standard, it is best that we do so ourselves as well. There are too many users (everybody using a typewriter or PC/computer). It is impossible to bring them together so that they could agree on a new standard and commit themselves to switch.

30.2.2.4.1 Inefficiency

The last consequence of positive feedback we mention is what is called possible inefficiency. This means that the best solution will not necessarily win. An illustrative and well-known example of this phenomenon is the competition between the Microsoft Windows operating system and Macintosh. Macintosh was widely held to be the best technology—in particular from a user point of view—but Windows won because it had early succeeded in building a large installed base.

30.2.3 World Risk Society and Reflexive Modernization

Ulrich Beck provides an extensive analysis of the essence and impact of globalization processes. IIs, such as the Internet, are key elements in globalization processes at the same time as globalization generates increased demands for more globally distributed and integrated IIs. The key element in Beck’s argument is that globalization radically changes our possibilities for controlling processes in nature and society, and then also the conditions and possibilities for managing IIs. The reason for this is, in short, the growth in complexity produced by globalization.

Beck builds his theory of (World) Risk Society on a distinction between what he calls first and second modernity. First modernity is based on nation-state societies, where social relations, networks, and communities are understood in a territorial sense. He argues further that the collective patterns of life, progress and controllability, full employment, and exploitation of nature that were typical in this first modernity have now been undermined by five interlinked processes: globalization, individualization,
gender revolution, underemployment, and global risks. The kind of risks he primarily focuses on is those related to the ecological crises (climate change) and global financial markets.

At the same time, as the risks are becoming global, their origin is also globalization (Beck 1986, 1999, Giddens 1990, Beck et al. 1994). And, further, both on-going globalization processes as well as the outcome—that is, increased risks—are fundamentally related to modernity. Globalization is the form modernization has taken today, and risks are increasing because of the changing nature of modernity. The very idea of controllability, certainty, and security—which was so fundamental in the first modernity—collapses in the transfer to second modernity (Beck 1999, p. 2). Modernity’s aim has been to increase our ability to control processes in nature and society in a better way through increased knowledge and improved technology. The concept of “world risk society” draws our attention to the limited controllability of the dangers we have created for ourselves. The reflexive modernization argument (at least in Beck’s version) says that while it has been the case that increased modernization implied increased control, in second modernity, modernization, that is, enhanced technologies and an increase in bodies of knowledge, may lead to less control, that is, higher risks.

This is what lies at the heart of the “reflexivity” argument. In particular, the theory of reflexive modernization contradicts the instrumental optimism regarding the predetermined controllability of uncontrollable things: “the thesis that more knowledge about social life... equals greater control over our fate is false” (Giddens 1990: 43), and “the expansion and heightening of the intention of control ultimately ends up producing the opposite” (Beck et al. 1994: 9).

This shift, which may appear contradictory, can be explained by the ubiquitous role of side effects. Modernization means integration. At the same time, all changes and actions—new technologies introduced, organizational structures and work procedures implemented, and so on—have unintended side effects. The more integrated the world becomes, the longer and faster side effects travel, and the heavier their consequences. Globalization, then, means globalization of side effects. In Beck et al.’s (1994: 175, 181) own words: “It is not knowledge but rather non-knowledge that is the medium of reflexive modernization... we are living in the age of side effects... The side effect, not instrumental rationality, is becoming the motor of social change.”

The theories of Risk Society and Reflexive Modernization can be seen as complexity theories in the sense that complexity is at their core. Seeing globalization as an integration process, for instance, can very well lead to it also being seen as a process of making the world more complex exactly through this integration process. The role attributed to side effects also links naturally to complexity. The role of side effects is increasing exactly because the “system” is becoming so complex. Because of its complexity, we cannot know how all components are interacting; accordingly, the outcomes of those interactions will be more unpredictable.

### 30.3 Design Challenges and Management “Tools”

#### 30.3.1 Design Dilemmas

##### 30.3.1.1 Tension between Standards and Flexibility

Infrastructures are made up of a huge number of components. Accordingly, standards defining the interfaces between components are essential features of IIs, and the specification and implementation of standards are important activities in the establishment of IIs. Standards are closely associated with stability. This is the case partly because keeping a standard stable is required for various reasons (e.g., data standards that make it possible to store data today and read them at a later time). But standards are also stable because widely diffused standards are so hard to change when we have to (because of the lock-in problem). However, standards need the change over time. Just like IIs, IIs need to change and adapt to changing user requirements. And some such changes mean that implemented standards need to change, too. At the same time, the scaling and growth of an II can generate need for changes in standards even though user requirements stay unchanged (Hanseth et al. 1996, Tilson et al. 2010).
Accordingly, IIs will evolve as a dynamic driven by a tension between standards (stability and uniformity) and flexibility (change and heterogeneity). Managing this tension is a key to the management of IIs (Hanseth et al. 1996).

### 30.3.1.2 Bootstrapping—Adaptability

The dynamic complexity of IIs poses a chicken–egg problem for the II designer that has been largely ignored in the traditional approaches. On one hand, IT capabilities embedded in IIs gain their value, as explained earlier, by being used by a large number of users demanding rapid growth in the user. Therefore, II designers have to come up early on with solutions that persuade users to adopt while the user community is non-existent or small. This requires II designers to address head on the needs of the very first users before addressing completeness of their design or scalability. This can be difficult, however, because II designers must also anticipate the completeness of their designs. This defines the bootstrap problem of II design. On the other hand, when the II starts expanding by benefitting from the network effects, it will switch to a period of rapid growth. During this growth, designers need to heed for unforeseen and diverse demands and produce designs that cope technically and socially with these increasingly varying needs. This demands infrastructural flexibility in that the II adapts technically and socially. This defines the adaptability problem of II design (Edwards et al. 2007, Hanseth and Lyytinen 2010). Clearly, these two demands contradict and generate tensions at any point of time in II design (Edwards et al. 2007).

### 30.3.2 Management “Tools”

So what kinds of “tools” are available for managing the evolution of IIs—or for installed base cultivation? The answer given here is: process strategy, architecture, and governance regime. The rationale behind the focus on these three aspects is, first of all, the fact that these three “tools” are what development efforts are all about: the steps to be taken to develop some new technology (bottom-up or top-down, incremental/iterative or “big-bang,” evolutionary and learning driven or specification driven, etc.); the architecture and overall design of the technology (the modularization of a system determines how and how easy it may be maintained and modified); and how to govern, manage, and organize the effort. In addition, these “tools” have also been at the centre of extensive discussions and research on the evolution of the Internet. These three factors have been three aspects as key factors behind its success: an experimental bottom-up development strategy (which includes a strategy and rules for bottom-up development and settlement of standards), the end-to-end architecture, and distributed control and governance structures combined with open source software licenses (Hanseth et al. 1996, Benkler 2006, Zittrain 2006).

Traditionally, the management of ICT has focused on the management of projects developing ICT solutions. Such projects are typically organized as a hierarchy of subprojects each with a subproject manager. Each manager has the rights to make decisions within the domain of the subprojects and give instructions to the managers at the level below. The management of such a project organization is normally supported by various management tools such as the detailed plans and establishment of milestones. And further, the production of detailed plans and the monitoring of the progress made in the project—if it is progressing according to the plan or not—are supported by various computer based project management tools. Together this package of project organization, decision rights, and management tools is an example of what I call governance regime. Governing the complexity of IIs requires new and different governance regimes. In the case of the Internet, for example, its successful evolution has been shaped by a governance regime consisting of a few central institutions like IETF and ICANN. Another important part of this regime has been the fact that most of its technology has been distributed based on open source license (like the GNU Public License). But may be the most important elements of the governance regime has been the organizing of the development activities as a loosely connected network of individuals that coordinate their work through extensive use of e-mail and by making all software and relevant information publicly available on FTP and Web servers.
Software and IS development often takes place by following specific methodologies. The central element of such methodologies is a specification of which steps to be taken, and in which sequence, to develop a specific IS solution. This is what I call a process strategy. The complexity of IIs requires process strategies different of those prescribed for traditional IS development efforts. In particular, we need process strategies that address the role of network externalities and path dependence; that is, we need strategies that address the bootstrap and adaptability problems.

The architecture of an IS is traditionally considered important for its maintenance. In general, modularization is an important strategy for coping with complexity. And in the case of IIs, the architecture plays a crucial role. This is illustrated with the role attributed to the Internet’s architecture in explaining its successful evolution. The Internet’s so-called end-to-end architecture (the functionality is located in the ends of the network, i.e., in the computers connected to the Internet, and not in the network itself, which has been the case within traditional telecommunication networks) has made the Internet extremely flexible in the sense that anybody having a computer connected to the Internet could develop and provide new services.

The management of IIs, then, requires process strategies, architectures, and governance regimes that in combination make an II evolve along the desired path. Exactly which combinations that is appropriate for specific IIs is still a major research issue.

30.4 Three Examples of Evolving IIs

I will here present three examples of IIs. They are all from the health care sector. They are presented as individual and separate IIs, but in reality, they will be linked together and with others and, accordingly, be parts of larger scale IIs in the health care sector. These examples illustrate the variety among IIs and, most importantly, how specific process strategies, architectures, and governance regimes are interacting and shaping the evolution of IIs.

30.4.1 Electronic Patient Records

30.4.1.1 Case Presentation

The introduction of electronic patient record (EPR) systems in hospitals is intended to improve the quality of patient care by replacing the existing fragmented and often unavailable paper-based patient records by an electronic one which would make any information instantly available to anybody, anywhere, and anytime. In 1996, the Medakis project was established with the aim to develop and implement an EPR system in all the five largest hospitals in Norway. Siemens was involved to take care of the software development. The system was given the name DocuLive. We focus here on the implementation at “Rikshospitalet” (see Hanseth et al. 2006), where DocuLive was intended to serve several ambitions: It should include all clinical patient information, covering the needs of all users; it should be built as one single integrated system; it should enable better collaboration and coordination of patient treatment and care through electronic information sharing and exchange. Finally, a more general and important aim, in the contexts of the arguments developed in this paper, was that the system should be a standard EPR solution for all Norwegian hospitals. This was an important goal for two reasons: enable information exchange and making it possible for health care personnel to use the EPR system without training when they are moving from one hospital to another. The deadline set for the delivery of the final system was the end of 1999. The project started with the best intentions of involving users, acknowledging current work practices, and favoring a bottom-up development strategy. However, as we illustrate in the subsequent sections, the difficulties associated with the standardization of both technology and work practices were dramatically underestimated.

Shortly after it began, project members became aware that within Siemens EPR projects were also underway in Sweden, the United Kingdom, Germany, and India. Moreover, they realized that the Norwegian project was not at the top of Siemens’ priorities since Norway represented the smallest market.
This meant there was a high risk of overrun as Siemens prioritized more profitable markets. As a consequence, the project members decided to make moves toward "internationalizing" the project, first to the Scandinavian level and later to the European one. However, this decision weakened the consortium's position with respect to Siemens, since now the requirements from all projects from across countries would need to be merged and a new architecture designed.

In 1999, Siemens acquired a large U.S. software company developing software solutions for health care. As a consequence, Siemens’ medical division’s headquarters was moved from Europe to the United States, and the project's scope became global. As the number of involved users grew, large-scale participatory development became unmanageable. After a few years, only a small number of user representatives from each hospital continued to actively participate in the development. Moreover, the need to continuously find common agreements between the hospitals turned the intended bottom-up approach into a top-down one.

Also, the efforts aimed at solving the fragmentation problem with a complete and smoothly integrated EPR system turned out to be more challenging than foreseen. Paradoxically, the volume of paper records increased, and the patient record became even more fragmented for a variety of reasons. First, this was because new laws on medical documentation required detailed records from professional groups not previously obliged to maintain a record (such as nurses, physiotherapists, and social workers). Second, for both practical and legal reasons, the hospital had to keep updated versions of the complete record. As long as lots of information only existed on paper, the complete record had to be paper-based. Thus, each time a clinical note was written in the EPR, a paper copy was also printed and added to the paper record. Printout efficiency was not a design principle for the current EPR, causing non-adjustable print layouts that could result in two printed pages for one electronic page form. Third, multiple printouts of preliminary documents (e.g., laboratory test results) were often stored in addition to final versions. The result was that the volume of paper documents increased. This growth created a crisis at the paper record archival department. The hospital had moved into new facilities designed with a reduced space for the archive as it was supposed to handle electronic records only. In 2003, the archive was full, and more than 300 shelf meters of records were lying on the floors. This situation also affected the time needed to find records, and often requests failed to be satisfied.

When the implementation of DocuLive started, five local systems containing clinical patient information existed. The plan was to replace these with DocuLive so as to have the EPR as one single integrated IS. In spite of this, the number of local systems was growing at an accelerating speed, based on well-justified needs of the different medical specialties and departments. For example, the in vitro fertilization clinic needed a system that allowed them to consider a couple as a unit, as well as allow tracking of information from both semen and egg quality tests through all procedures involved, up to the birth of the child. The intensive care unit acquired a system that allowed them to harvest digital data from a vast array of medical equipment and thus eliminate the specialized paper forms previously used to document events and actions. Moreover, new digital instruments in use in many different departments include software components with medical record functionality. The number of such specialized systems had grown from 5 in 1996 to 135 in 2003.

The original plan said that the final system should be delivered in 1999. Four years later, toward the end of 2003, the version of DocuLive in use included information types covering between 30% and 40% of a patient record. This meant that the paper records were still very important, but unfortunately more fragmented and inaccessible than ever. The increased fragmentation was partly due to not only the large volume of paper caused by DocuLive printouts but also the high number of specialized systems containing clinical patient information.

At Rikshospitalet they decided to change their strategy and approach complexity in quite different way in 2004. They realized that the idea of one complete EPR system had failed. Instead, they decided to “loosely couple” the various systems containing clinical patient information underneath a “clinical portal” giving each user group access to the relevant information in a coherent way.
30.4.1.2 Another EPR Case

I will here briefly present another EPR case as a contrast to the aforementioned one. Around the same time as the Medakis project started, a small and simple application supporting a few work tasks for a couple of medical doctors was developed by a software developer at a smaller hospital in Northern Norway. The users were very happy with the system. After a period of use, they asked for some added functionality, which was implemented. At the same time, other users became aware of the system and wanted to try using it as a tool supporting some of their work tasks. This was the first iteration of a long evolutionary process. The functionality, number of users, and developers involved started to grow. After a few years, the software and the developers were taken over by a company set up for this task. The company and the software product was given the name DIPS. It has continuously been growing and is today standard EPR system in Norway (Ellingsen and Monteiro 2008). In September 2012, the Oslo University Hospital (which Rikshospitalet is a part of) decided to replace DocuLive with DIPS. This means that four of the five hospitals involved in the Medakis project will have abandoned DocuLive in favor of DIPS.

30.4.1.3 Case Analysis

How can the project trajectory of DocuLive at Rikshospitalet be analyzed? Key issues are the complexities involved and the handling of these complexities. The complexity of the Medakis project and the DocuLive system was made up of a mix of technical, organizational, and medical issues—that is, a very complex actor-network. Further, this case demonstrates the limitations of traditional strategies for managing software development projects and how these strategies lead to actions triggering lots of side effects. These side effects are propagating throughout the actor-network, creating domino- and boomerang effects, which are returning to the origin of the initial actions so that the final results became opposite to what one aimed for. In this way, the case demonstrates the relevance of the theory of reflexive modernization in the sense that strategies aiming at control of the development process leads to less control.

The primary complexity involved is that of the work practices related to patient treatment and care. By trying to make one integrated system that should cover the needs of all hospitals in Norway, one brings together the total complexity of these practices. The hospitals realized that dealing with this was demanding. Accordingly, they concluded that they had to involve a supplier that was strong both financially and in terms of ICT competence. So Siemens was chosen. But then, the complexity of Siemens was merged with that of the hospitals. The medical division within Siemens is large, with a traditional base within medical imaging technologies. The imagining instruments had become digital, and supplementary software systems had been built. As the EPR development activities were increasing within Siemens, it became more and more important to align and integrate the EPR strategy and product(s) with other Siemens products and strategies.

Within this world, Norway become marginal, as the appetite for larger markets escalated in a self-feeding process. A side effect of the expansion of ambitions and scope was the increased complexity: the larger the market Siemens was aiming at, the more diverse the user requirements, and accordingly, the more complex the system had to be in order to satisfy them. This implied that the development costs were growing, which again implied that a larger market was required to make the whole project profitable. The project went through this spiral of self-reinforcing and escalating complexity several times until it collapsed.*

The failure of DocuLive can be explained in more theoretical terms as a failure in attempting to control complexity. Arguably, the main mistake was to follow a “traditional” IS development approach—typical

* At Rikshospitalet, they decided in 2003 to change their strategy quite dramatically. In this new strategy, the central element was loose coupling of the various systems through a portal, which was giving various user groups coherent interfaces to the systems they needed to access. This strategy has been much more successful—but not without its own challenges.
Developing and Managing Complex, Evolving Information Infrastructures

for (first) modernity, that is, overemphasizing criteria of universality, uniformity, and centralization of control to achieve alignment, stabilization, and closure. In line with the more recent developments within actor network theory, our case data suggest that the complexity defines II development as the emergence of multiplicities, inconsistencies, ambivalence, and ambiguities (Law and Mol 2002, Law 2003b). Ironically, what happened became the opposite of the initial aims. When actors tried to stabilize the requirement and design specification by enrolling more actors, this made it less stable. Attempts to improve fragmented records by means of one integrated EPR made the records more fragmented. The complexity of DocuLive turned out to be one where the ordering efforts created disorders. The side effects triggered new ones, which again were reflected back on the origin—the standardization process turned out to be reflexive and self-destructive. The dynamics of reflexive processes at work are summarized in Figure 30.1.

The concept of reflexivity offers thus an interpretation of the dynamics of the case. The theory of “high” modernity helps to observe how the logics of the “first” industrial modernity find their limits (Beck et al. 1994). The intensified interconnectedness of social practices with technical artifacts on the one hand, and the need to align geographically dispersed actors on the other hand, effectively undermines the reductionist approach to control complexity. The weaknesses of such approach become visible when the control itself reflexively reproduced the complexity—thus creating the immanent paradox of modernity. This fact highlights the need to develop alternative standardization approaches that better overcome the paradoxes to deal with complexity.

The case is also an example of self-reinforcing processes driving the dynamics of complex systems. In this case, what the most salient self-reinforcing process is that of the escalation of the complexity.

I will now zoom in a bit more on the specific “management tools” chosen and their impact on the evolution of the DocuLive solution and the Medakis project. Most important is the principal architectural decision made: The solution should be a tightly integrated (or rather monolithic) one. On this basis, it was decided to organize the development activities as a classical hierarchical organization controlled from the top by a project manager and to develop the solution according to a top-down, specification-driven process. The choice of process strategy and governance regime was largely given by the choice of architecture. The important role played by the architecture is illustrated by Rikshospitalet’s decision to switch to the almost exact opposite strategy in 2004, that is, an architecture allowing the implementation and use of a large number of applications loosely coupled through a portal. This strategy, then, implied a more evolutionary approach (i.e., process strategy) to the development of individual

FIGURE 30.1 The reflexive standardization process.
applications as well as the overall II, and this evolutionary process was managed by means of a network oriented governance structure in terms of a number of loosely coupled projects. This strategy turned out to be much more successful, and the paper-based patient record was replaced within a couple of years.

The history of DIPS is almost the exact opposite of the Medakis project and DocuLive. It started out as a very small and simple solution design to support a very narrow range of works tasks and very few users. The very first step taken was successful. Learning from and capitalizing on this, the solution has been growing into a national Norwegian EPR II. This process strategy illustrates very well how IIs may bootstrap: as the installed base of an II grows, the use value of the II increases which, in turn, attracts more users in a self-reinforcing spiral. The architecture of DIPS, however, shares its basic features with DocuLive: it is a tightly coupled system. But the bottom-up evolutionary process strategy chosen made it possible to make the solution over time grow into one satisfying the required range of user requirements of a national II. The governance regimes have some common features (proprietary software delivered by a commercial company), but are also different in important areas. The Medakis project was set up as a national and hierarchical organization including representatives from each of the five hospitals. The DIPS governance regime had a more entrepreneurial flavor. The strategy for how a DIPS-based II should grow toward a national one has been determined by the company. User organizations have primarily been involved as customers and as individual hospitals (later regional companies) related to individual implementation projects at each hospital.

### 30.4.2 Information Exchange between Health Care Institutions

#### 30.4.2.1 Case Presentation

The second case presented is an example of an II supporting information exchange between health care institutions involved in diagnosing and treatment of patients. We will look at one such project, called the Elin project, which aimed at establishing electronic co-operation between the general practitioners (GPs) and other actors in the healthcare sectors, such as hospitals, pharmacies, and welfare authorities.

The aim of the project was first to develop comprehensive requirements specifications as a basis for user friendly, standardized solutions for electronic healthcare-related communications for GPs. The main aim was “better communication and collaboration, and not just development of technical solutions for message exchange.” This included the development of solutions for exchange of admission and discharge letters, laboratory orders and reports, illness and doctor’s declarations, prescriptions, and communication with patients. The project had a strong focus on standards representing the main information objects, like laboratory orders and reports, admission and discharge letters, and prescriptions.

The project was split into three phases. In the first, the focus was on exchange of discharge letters between GPs and hospital departments and outpatient clinics. In the second phase, the focus was on exchange of discharge letters between medical specialists’ offices and GPs, exchange of orders and reports between radiology laboratories to GPs, and information exchange with patients. The third phase focused on improving and piloting the technical solutions. The II’s architecture was derived from what we can call the “EDI paradigm.” That means that the overall II would be built by adding functionality to the applications the users were using, like the GPs’ patient record systems for sending and receiving the specified messages. This means that all vendors of applications within the use area need to be involved and agree on the specifications and implementing them.

A number of challenges surfaced. For example, the existing EDIFACT standards did not fit well with the defined requirements, and for some messages, standards were not yet available. Another challenge was that the exchange of the messages took too long time for various (and often mysterious) reasons. Accordingly, new standards and messages had to be specified based on a web services model but within the framework of the EDI paradigm and its architecture.

The project has played a major role in the development of user requirement for ICT solutions supporting communication between GPs and other healthcare institutions, which are well aligned with user...
needs and requirements from healthcare authorities. The project also specified a standardized architecture for solutions for information exchange between GPs and hospitals (Aksnes 2006). It was quite successful in establishing strong, enthusiastic collaborative networks of users and suppliers. However, the implementation and diffusion of solutions have definitively been very, very slow. In spite of this, the approach taken by the Elin project has been considered a great leap forward compared to previous efforts within the same domain because it had a stronger focus on user requirements and less on specification of technical standards only. For this reason, it triggered a series of follow-up projects in related areas. But the success of these projects in terms of developing solutions that are adopted by the intended users is modest.

30.4.2.2 Case Analysis
The overall approach taken by the Elin projects is similar to Medakis, and so are the outcomes. But there are also differences. The overall complexity of the work practices one aimed at supporting is significantly lower. But the most important difference is the complexity of the organization developing the Elin IIs, that is, the overall complexity of the actor-network created by the chosen architecture. The organization of the development work was a direct side effect of the choice of architecture.

Even though the number of people involved was rather small, the project involved many different and totally independent vendor organizations. The way the development work was organized was a direct implication of the architecture chosen, that is, the EDI model where the information exchange is implemented as add-ons to and tightly integrated with the applications in use. This work can hardly be done by others that the applications vendors. So all of them needed to be involved, they had to agree upon quite complicated technical specifications, and each of them needed to implement the specifications “correctly.” The experience is that this requires tight coordination between a large number of independent actors—in fact, tighter coordination that what is possible.

30.4.3 Telemedicine in Ambulances
30.4.3.1 Case Presentation
The last case is the implementation of a telemedicine solution in ambulances. This is a successful case—largely due to the bootstrapping strategy followed (Hanseth and Aanestad 2003).

Østfold is the south-easternmost county in Norway, with a population of around 250,000 on an area of 3600 km². Until January 1, 1998, there were five independent public hospitals in the county, each with their own fully equipped emergency care unit (ECU). In a process of rationalization and centralization, the hospitals were merged into one organization. The number of ECUs was reduced to two, which created public concern over increased transport time and possible loss of lives in emergency situations. Also time spent within the hospital before treatment is started (“door-to-needle-time”) is generally longer in large than in small hospitals. For example, for myocardial infarction, treatment should preferably be given within an hour (“the golden hour”). Usually myocardial infarction is treated with thrombolytic agents that dissolve the blood clots; however, this should not be given in certain cases as it may cause massive bleedings. The procedure was thus to wait with thrombolytic agents until the patient was brought form the ECUs to the heart intensive care unit and a doctor had verified the diagnosis of myocardial infarction using electrocardiography (ECG) equipment. This means that in many cases the patients do not reach doctors that can give this important treatment within an hour—unfortunately with serious consequences.

In 1996, prior to the merger, one doctor had heard of MobiMed, a telemedicine systems that facilitated transmission of text and ECGs from ambulances to a receiver (e.g., in a hospital). He and some ambulance drivers went to look at the system in use in Sweden. When approaching the county’s health administration, the doctor did not manage to convince them to support this financially, but he got a permission to try out the system. The vendor lent the equipment, and in February 1998, two senders were
installed in two ambulances, and a receiver was installed in the cardiology ward in the hospital where these ambulances would bring their patients. The first aim was to transmit ECG to the cardiology ward for interpretation by a doctor. A verified diagnosis of myocardial infarction would allow the ambulance personnel to bypass the ECU and bring the patient directly to the heart ICU.

The ambulances in this town were selected because there was here a rather high likelihood that there would be an anesthesia nurse available for emergency trips. Anesthesia nurses were trained in using the equipment, and the practical testing started. During 1998, it was used on 166 patients; of these, 16 had an infarction. In these cases, the ECU was bypassed and the “door-to-needle time” was reduced between 25 and 30 min. The ambulance personnel were also taught how to use an ECG recorder, and they practiced on each other during quiet time periods. After some time, they were also allowed to use the equipment when there were no nurses in the ambulances.

In January 1999, the ambulance nurses were allowed to administer the thrombolytic medication (i.e., giving the patient the shot) after the diagnosis was verified by the doctor on duty at the hospital, and the total time from the patient became ill to the medication was given (“call-to-needle-time”) was further reduced by 25–30 min. Sometime later the ambulance personnel were also trained and allowed to administer the medication.

Based on the success with the first two ambulances and the benefits and savings that they could demonstrate, the county’s health authorities decided to support the purchase of the equipment also for ambulances from the other towns that did not have an ECU. Here, the equipment was installed in April 2000, and the transition from just bypassing the ECU to also give medication in the car came about in just 6 months (October 2000), as compared to about 1 year in the first case.

During 1999, the Norwegian health authorities were approving the practice where nurses were administering the medication after a verified diagnosis. Up to that time, they could do it as a task delegated by the doctor setting the diagnosis. Such an approval from the authorities was necessary to scale up the activities. Later on, a similar authorization was given to the ambulance personnel and how to do it was included into their regular education.

By December 2001, it was decided to purchase senders also for the rest of the ambulances, including the towns with an ECU. The aim was then not to administer medication in the ambulance but to bypass the ECU when appropriate. Similar projects were started in several other counties and the extension of the system for also supporting the diagnosis of brain stroke was under discussion.

30.4.3.2 Case Analysis

This case demonstrates, first of all, that substantial benefits may be obtained with limited resources when an appropriate strategy is chosen. In the two previous cases, the chosen strategy generated an unmanageable complexity. Like the DIPS case, this one followed a strategy keeping the complexity at lower level. But also this case was complex in the sense that it included a rather complex mix (actor-network) of technological, medical, and legal issues. First of all, this case demonstrates the importance of an evolutionary and learning centered approach, which balances short-term gains and long-term strategic aims. This evolutionary strategy was crucial in order to prove that it was possible for first nurses and later on other ambulance personnel to give the shots to patients in a safe way. This was achieved within a framework of collaborative arrangements between personnel in the ambulances and cardiologists in the hospital that were developed over time in combination with the overall II that was established. These collaborative arrangements were developed in tandem with new rules and regulations that made these arrangements legal at the same time as they both (regulations and collaborative arrangements) were based on solid practical experience proving them safe.

The evolutionary process strategy chosen was made possible by key features of the technological architecture: a very simple technological solution, which was loosely coupled to the larger infrastructure in terms of the existing equipment in the ambulances and the EPR solution and overall IT infrastructure in the hospitals. Lastly, the strategy was also supported by the entrepreneurial governance regime:
The II evolution was primarily driven by a few enthusiastic doctors. This lean governance regime also contributed to making the smooth evolution of the II possible.

30.5 Conclusion

Overall the complexity of ICT solutions, individually and through integration with others, has reached a level of complexity that demands radically new approaches to the way we develop and manage them. Such new approaches also need to be built on new concepts. In this chapter, the concept of IIs is used in order to capture the complexities ICT developers and managers are confronted with, the key characteristics of these complexities, and the challenges they raise regarding their development and management. The chapter also points to some ways of coping with these challenges.

The key characteristic of IIs is, of course, their complexity. Complex systems are evolving in a way largely driven by side effects. So managing IIs needs to focus on these side effects—how to avoid creating them but at the same time draw upon them. In addition to their complexities, the key characteristic of IIs is the fact that they are not designed from scratch and that they do not have a life cycle—they just evolve. So approaches to coping with their complexities need to focus on how to shape their evolution—cultivating the installed base. In this chapter, three “tools” are presented, which may help us in managing complex IIs: process strategies, architectures, and governance regimes. Managing IIs, then, requires the choice of an appropriate combination of these tools. We have illustrated how some such combinations shape the evolution of three IIs within the healthcare sector. But we have still a long way to go before we have the “toolbox” required for proper management of IIs. Unfortunately, we might never reach that stage because as we are developing better “tools,” the complexity—and the challenges for management—is growing. Substantial research into this critical issue is desperately needed. In my view, what is most in demand is research on how specific combinations of the management “tools” mentioned here in interaction shape the evolution of IIs. Based on such research, better advice can be given on how specific “tools” should be selected for specific IIs.

References


