Volume 4 Issue 4 Article 12 - 2004 **ISSN 1535-6078**

Sprouts: Working Papers on Information Environments, Systems and Organizations

Theorizing about the Design of Information Infrastructures: Design Kernel Theories and Principles

Ole Hanseth
Oslo University

Kalle Lyytinen
Case Western Reserve University

Abstract

In this article we theorize about the design of information infrastructures (II). We define an information infrastructure as *a shared, evolving, heterogeneous installed base of IT capabilities based on open and standardized interfaces.* Such information infrastructures, when appropriated by a community of users offer a *shared resource* for delivering and using information services in a (set of) community. Information infrastructures include complex socio-technical ensembles like the Internet or EDI networks. Increased integration of enterprise systems like ERP or CRM systems has produced similar features for intra-organizational systems. Our theorizing addresses the following challenge in designing information infrastructures: how to tackle their inherent complexity, scale and functional uncertainty? These systems are large, complex and heterogeneous. They never die and evolve over long periods of time while they adapt to needs unknown during design time. New infrastructures are designed as extensions to or improvements of existing ones in contrast to green field design. The installed base of the existing infrastructure and its scope and complexity influence how the new infrastructure can be designed. Infrastructure design needs to focus on installed base growth and infrastructure flexibility as to avoid technological traps (lock-ins). These goals are achieved by enacting design principles of immediate usefulness, simplicity, utilization of existing installed base and modularization as shown by our analysis of the design of Internet and the information infrastructure for health care in Norway.

Keywords: Design Theory, Information Infrastructure, Evolution, Validation, Internet, Business Infrastructures.

Citation: Hanseth, O., and Lyytinen, K. (2004), "Theorizing about the Design of Information Infrastructures: Design Kernel Theories and Principles," *Sprouts: Working Papers on Information Environments, Systems and Organizations*, Volume 4, Issue 4, pp 207-241. http://sprouts.case.edu/2004/040412.pdf

Theorizing about the Design of Information Infrastructures: Design Kernel Theories and Principles

Introduction

The history of information technology (IT) deployment has been characterized by a steady increase in the flexibility, reach and range of computer based information services (Keen 1991). These have been driven by radical improvements in computing powered, lowered cost and enhanced software capability. As a result separate information systems (IS), system functionalities and software tools have over time become integrated into complex ensembles of heterogeneous IT artefacts, which are increasingly connected with and dependent upon one another. Such a complex, evolving and heterogeneous socio-technical system we call here an *information infrastructure* (II). We define an information infrastructure as *a shared, evolving, heterogeneous installed base of IT capabilities among a set of user communities based on open and/or standardized interfaces.* Such an information infrastructure, when appropriated by a community of users offers a *shared resource* for delivering and using information services in a (set of) community. Internet or industry wide EDI networks are examples of large, successful information infrastructures. We see traditional information systems to be transformed by their advances in reach, range and integration into complex corporate wide and industry wide information infrastructures (Broadbent and Weill 1998). We regard these information infrastructures a new class of IT systems which need to conform to a different set of design requirements than traditional information systems (Walls et al 1992).

Despite significant past research on information system design and strong advances in formulating effective design strategies for many types of IS (see e.g. n. Walls at al. 1992, Fitzgerald 2000, Mathiassen et al., 2000) we currently have a dearth of knowledge how to effectively design information infrastructures. This is surprising given the fact that significant challenges in recent years in IS management and design are directly or indirectly related to how to manage complex and heterogeneous infrastructures (Ciborra 1996, Weill and Broadbent 1998,Lederer and Salmela 1996). IIs possess several characteristics which make their design different from design tasks faced in the past (see e.g. Markus et al. 2002, Jones et al. 2003, Walls at al. 1992): 1) IIs are large, complex and evolve over a heterogeneous set of communities and components. 2) IIs need to adapt to both functional and technical requirements that are unknown during design time. 3) Third, IIs are commonly designed as extensions to or improvements of existing ones and they combine and draw upon heterogeneous and diverse components that are not under the control of a designer. As a result II design must recognise how the installed base of the as-is infrastructure and its scope and complexity influences the on-going design.

In this paper we outline a design theory (Walls et al. 1992) for information infrastructures. We discuss ingredients of the "kernel theory" of information infrastructures upon which our understanding of the effective II design can be based. The proposed kernel theory draws upon our field work, and an appraisal examination of complexity theory (Cilliers 1996), evolutionary economics (Arthur 1988 1992, David 1986, David and Bun 1988, Katz and Shapiro 1985) and social shaping of technology research (Bowker and Star 1999, Latour 1991, 1999) for design theory. In consequence we formulate a set of design guidelines as Simonian procedural rationality (1981) derived from this kernel theory and validate them through case studies that focus on Internet evolution and health care infrastructures

The remainder of the article is organized as follows. In section 2 we define IS design theory and review past research on IS design theories. Section 3 reviews past research on information infrastructures and offers a note on research methodology. In section 4 we develop the first part of the kernel theory of II by defining critical features of IIs (their set of system features). In section 5 we define design goals and key theoretical principles related to the evolution of IIs. In section 6 we formulate guidelines for II design. In section 7 we illustrate how these principles were followed during Internet design. In section 8 we discuss how these guidelines were not followed in developing a failed industry wide vertical infrastructure for health care. Concluding remarks and ideas for future research follow in section 9.

Design Theories in the IS Field

Introduction to Design theorizing

Since the publication of Walls et al (1992) the term "IS design theory" has been used in a specific sense. It refers to a set of concepts, beliefs, conjectures and generalized scientific laws (both natural and social) by which designers map effectively design problems to solutions for a special class of IS problems. Normally these classes are identified by specific labels as DSS, EIS, or EKP (Markus et al 2002). Such bundles of knowledge encapsulate and organize three interrelated elements: 1) a set of requirements for a family of design problems, 2) a set of system features (or a set of principles selecting system features) that meet these requirements, and 3) a set of principles deemed effective for guiding the design process so that a set of system features is selected and implemented that meets a given set of requirements (see table 1). By addressing all these elements simultaneously the IS design theory bundles together a set of guidelines justified by theoretical warrants that can offer effective guidance for designers facing a set of design contingencies.

The notion of design theory as a package of beliefs, norms and theoretical concepts underscores two distinctive characteristics of design: 1) each design draws upon theory-in-use and by doing so the quality of the resulting design validates this theory when the application of design principles has been successful, and 2) it offers effective normative guidance to practice by formulating rules that are general and thus applicable over a set of design situations (Lyytinen 1987).

Requirements/ goals	Describes the class of goals to which the theory applies.
A set of system features	A set of IT artifacts (class) hypothesized to meet the requirements.
Kernel Theory	Theories from natural and social sciences governing the design requirements or the processes arriving at them.
Design principles	A codification of procedures which when applied increase the likelihood of achieving a set of system features. These procedures are derived logically from kernel theories.

Table 1. Components of a IS design theory

The first element- theory- is referred as "kernel theory" (Walls et al 1992). This can be either an academic and scholarly formulated set of concepts, statements, or practitioners' "theory-in-use" which is made explicit through hermeneutic process of codification. A kernel theory enables to formulate testable predictions of a class of solutions and their behaviors, or the associated design process. An example of such a testable prediction would

be: when designers apply controlled abstraction it leads to better quality software than when such principles are not used (Parnas 1972). The internal validity of a design theory can be improved when theory builders observe design outcomes in situations where the design theory was enacted and contrast them to situations where it was not. This is, however, difficult to achieve due to sampling and measurement problems, or the impact of history, context, and learning for design outcomes. All these negate the possibility for evaluating in a controlled fashion design theory treatments (Fitzgerald 1991). The value of kernel theories is to stimulate research on classes of IS, their properties and structural features as to offer grounded warrants for specific future IS design.

The value of design theories for IS research and practice is twofold. First, like all good theories design theories embed abstractions which enable effective transfer of practical knowledge to new situations (external validity). By doing so they reduce designer's uncertainty in design by effectively limiting 1) the design search space into a set of manageable options, and 2) by improving the search quality and speed. This increases design reliability and reduces variance and thereby increases the likelihood of success when evaluated in light of design goals. Second, IS design theories are normative- to follow Marx' famous slogan they do not explain the world (only), but seek to change the world. Good design theories state effectively how one should behave, or what is good or bad for a given design situation. They embed prescriptive and evaluative elements (design ideals) into design practice and justify them in relation to desired outcomes (Hevner et al 2004). Thus, design theories must pass in addition to concerns of internal and external validity the verdict of practice: Can these principles be effectively applied?

Review of Existing Design Theories

There are few explicitly formulated design theories within the IS field that meet all the requirements stated above for a good design theory. Walls et al (1992) offer relational design theory as one classical example. Most design theories are steps towards improved *theorizing* about a set of design problems, and they meet only partially all the requirements defined above for a design theory (Weick 1995). Incomplete or speculative design theories, as Walls et al. (1992) notify, can be formulated for any stage (or situation) for any class of information systems. A union of such sub-theories would result in a completely general design theory which would cover any stage or design situation for any type of IS. Such theory does not currently exist and we doubt the value or even possibility of such theory. To build a completely general IS design theory, which is sufficiently accurate as to help formulate effective design guidelines would become a too complex undertaking for any aspiring theory builder. For example, its validation alone would establish insurmountable challenges. Moreover, it would be too wieldy to use for any practical situations (Weick 1986) thus negating the very purpose of a design theory. Therefore, for both theoretical and practical purposes all available design theories have been formulated in view of a limited class of design problems.

The most common approach in defining more limited design theories has been to apply either vertical or horizontal criteria by which theory builders can narrow down to classes of design problems. Each design theory is thereby assumed to apply only in a certain context in which a specific set of requirements apply to a specific class of information systems (vertical demarcation), or to specific class of design problems across systems (horizontal demarcation). Any *practical* design engagement will, however, involve mobilization and enactment of multiple design theories as specific bodies of knowledge that apply to IS design (Hirschheim and Klein 2004).

Vertical design theories embed domain specific design theories formulated for a specific class of information systems that share unique properties or structural features. In

line with this scholars have brought forward design theories for decision support systems (DSS), executive information systems (EIS) (Walls et al 1992), transaction processing systems (TPS) or emergent knowledge process (EKP) systems (Markus et al. 2002), among others. A vertical design approach on a higher level is also a set of Information systems planning approaches which seeks to address the problem of how to specify and manage a collection of ISs *before* designing any single IS (Lederer and Salmela 1996).

Horizontal design theories address first general process features across a set of design situations. It is assumed that some process features are instrumental in arriving at a good solution no matter which type of system one works with. Examples of such process design theories are IS life cycle models which offer a widely accepted design theory for effective process organization (Walls et al 1992), contingency models which suggest ways choose between process approaches in alternative design context (Davis 1982), or process theories that predict interactions between changes in the amount of people working on design and its delivery time (Brooks 1968). Second, horizontal design theories can be geared towards specific design contexts that are common across classes of information systems[1]. Examples of horizontal design theories focusing on specific design situations include relational data base theory (Codd 1972), principles of information hiding and encapsulation (Parnas 1972), or critical success factor approach (Rockart 1979) which was originally developed in the context of EIS design but has become widely applied across different types of IS design situations (Walls et al 1992).

A key assumption behind all design theories we have reviewed is this that the final design will be generated from scratch by matching identified context specific user requirements (obtained through and assumed in the process kernel theory) with goals/ requirements of a given vertical and specific horizontal design theory. The un-stated assumption here is that a majority of requirements underlying design can be specified beforehand and the final design problem is assumed - for all practical purpose - to be a stand-alone closed artefact that follows a specific life-cycle. In this regard, the kernel theories do not regard the design artefacts' or design communities' relationships with other artefacts in the current, or to-be design environment as critical[2]. We find these assumptions untenable in relation to a set of design problems that one faces wit the design of information infrastructures. As noted above IIs evolve and have longevity beyond designers' own time-frame, and therefore do not yield to life-cycle concepts. The II design boundaries and elements are only partially known and controlled by the designer, and there can be multiple designers. The design never starts in a green-field situation, and therefore design requirements cannot be solicited unambiguously. In contrast, the installed based acts like an actor that sets up requirements, too. To our knowledge no design theory of information infrastructures has been developed that addresses these concerns. Accordingly, we seek to theorize about a new kind of (vertical) IS design theory of information infrastructures which assumes that considering specific infrastructure features is instrumental in their design.

[1] It is interesting to note that most vertical theories have been developed by IS scholars but most horizontal theories originate from computer science or software engineering communities.

[2] For example, a standard text book on object oriented methodologies (Mathiassen et al., 2000) spends exactly 1% (4.5 of 450) of its pages on systems integration and issues how how to relate design to-be with the current environment.

Related research and Research Methodology

Related Research

In the past some research has been carried to overcome limitations of theorizing about the design of complex IT infrastructures. Recent studies to explain information system change view system evolution as an instance of organizational improvisation (Ciborra 1996, Orlikowski 1996). This theory rests on two assumptions which differentiate it from traditional IS design: 1) system changes succeeding a technology implementation constitute an ongoing development process. Thus, after the implementation the design is still ongoing; 2) technological and organizational changes that take place during the organizational adoption (=improvisation) cannot be anticipated ahead of time- hence the design is blind and takes place behind the veil of time-space disjuncture. Assuming that a design can start with a stable and complete set of requirements is moot. These assumptions are similar to what we characterized of infrastructure. This theory, however, does not offer sufficiently strong basis alone for formulating a kernel design theory. First it focuses solely on user driven design when the infrastructure is already in place and ignores the active role of designers in installing it. Second it downplays the complexity and scale of technology choices that are involved in the design of information infrastructures. A similar attempt is made in Porra's (1999) work on colonial systems. She views the design process as some type of situated improvisation, but in contrast, focuses on design as a collective endeavour. In particular, she underscores the role of history and the inherently path-dependent nature of design: past experience and action i.e. the state of the infrastructure and its installed base shapes always future design. This aspect makes her concept of design close to the design theory proposed below. Her discussion, however, does not address how longevity, large scale and complexity can be addressed in infrastructure design.

What comes closest to our attempts to theorize is research on enterprise wide information infrastructures (Weill and Broadbent 1998). The main contribution of this line of research has been managerial models that help estimate the value of an enterprise wide infrastructure (a value based kernel theory for infrastructure design) and formulation of strategies how to utilize such infrastructures for value creation (design principles). The value driven model views an infrastructure as an IT portfolio, similar to any other investment portfolio. The approach is valuable in understanding how infrastructure decision can be made from a managerial perspective. It also takes into account the need for continuous revision of the infrastructure and the impossibility to predict all design requirements at the design time. The view, however, has limitations for an II design theory. First, it views infrastructure only through a financial lens and regards it as a portfolio of investments. This maintains a stark split between an application and an infrastructure the research does not address of how to actually go about designing large evolving applications or to integrate them. Second, the investment portfolio analogy is useful only to understand how to govern financial yields of information infrastructures but not how design them. In design situations the analogy is misleading and even dangerous. Investment portfolios are flexible, liquid and easy to value, control, and change. All II's as we know them have none of these features. Their elements are highly interdependent and their configurations are brittle so buying and purging infrastructure components cannot be done freely. Moreover the complexity of information infrastructures makes them extremely difficult to value and control.

Research Methodology

The proposed theory of II design summarizes and condenses more than 15 years of authors' research and development around information infrastructures. The research started

in the late 80's when one of the authors got involved in developing information exchange standards for the health care sector. These developments were not successful and the following research then focused on explaining this and other observed failures related to II design. One means to address this challenge was to investigate available accounts of other infrastructure developments like how building of railroads or electric power grids either succeeded or failed (Latour 1993, Hughes 1987). Another was to explore examples of successful II designs and contrast them with the failed ones. To this end we scrutinized the evolution of the Internet and solicited explanations, which both the original developers of Internet and its researchers have offered to explain its success.

The design theory outlined below originated from our attempt to contrast the success of the Internet with the failure of the health care initiatives we had personally experienced by relating these findings with explanations of large scale infrastructure initiatives. The research approach followed in formulating the design theory was based on methodological triangulation. We used action research and reflectively explored our experiences with failed II efforts to find out reasons for failures. Second, we carried out a careful literature review to learn from other infrastructure research in formulating the kernel theory and design principles. Finally, we used two case studies as a means to derive testable hypotheses from our theory and validate them for internal and external validity (Yin 1994).

Kernel Design Theory: Critical Features of Information Infrastructures

We will next present the key elements of design theory of II using the model as suggested in table 1 as a basis. The resulting key features of design II theory are summarized in table 2. These features will be presented in more detail in sections 4, 5 and 6. In this section we will define the system features that are shared by successful information infrastructures by defining our unit of our analysis – information infrastructures- and its interaction principles. This ensuing analysis offers the basis to formulate a kernel theory of information infrastructure evolution in section 5.

Definition of an Information Infrastructure

To articulate the kernel theory we need to first define our basic unit of analysis: what is an information infrastructure? The definition needs to highlight what makes IIs as units of analysis and as targets of design different traditional information systems so that we need new kernel design theories. We define an information infrastructure as *a shared, evolving, heterogeneous installed base of IT capabilities among a set of user communities based on open and/or standardized interfaces.* Such an information infrastructure, when appropriated by a community of users offers a *shared resource* for delivering and using information services in a (set of) community.

First, the definition articulates a different concept of an IT artefact from the traditional definition of an information system. Traditionally an IS is seen as an application -aka user tool-, which is developed to serve dedicated organizational tasks. In contrast, information infrastructure has no specific purpose or goal that justifies its existence, other than a very general idea of offering information related services to community. In contrast the labels for classes of information systems like accounting systems, payroll systems, transaction processing systems, decision support systems, executive information systems portray clearly the dedicated purpose organizational task that is being supported. The specific task being supported is normally mandated by specific organizational roles and functions, and accordingly the system is used and appropriated by a well identified and limited set of users (e.g. accountants and secretaries in the accounting department or other parts of the

organization). If this group changes the system will change and loose its identity. If the II loses some or all its users and replaces them with others, it does not lose its identity.

Second, IIs *evolve* continuously and unexpectedly in that their boundaries are not fixed beforehand. Infrastructure evolution implies anticipation of a continuous change in the II's scale, scope and functionality. Due to this evolution information services and associated components in the information infrastructure will expand (or sometimes shrink) in time and space in an organic manner. This change does not necessarily relate to any specific plan or goal like with traditional information systems. The design requirements for IIs consequently differ from those with single ISs where the growth is predictable and locally bound. For example, internet has been growing exponentially in un-anticipated ways in its scale and scope since the early 80's. Its number of technological components has grown enormously as its number of users. At the same time its functionality and the set of embedded capabilities has changed fundamentally and unexpectedly through innovation that has introduced new protocols like http, HTML, or more recent standards like XML or SOAP (Tuomi 2001). In the similar manner in corporations families of applications where each one is integrated with at least one other application (i.e. the network is connected) have evolved continually in the sense that each application can be integrated with additional ones, while at the same time new applications appear and become connected. The fact that infrastructures evolve over a long periods of time and have no clearly defined boundaries in scale, scope and functionality has important implications in understanding *how* the evolution unfolds, and what kind of strategies can be adopted for the design of II. When a part of an infrastructure is changed or improved, each new feature, or each new version of a component has to fit with the as-is infrastructure. This as-is infrastructure – i.e. its *installed base* - and its organization heavily influences how a new infrastructure or its part can be designed, and, in fact, how it can evolve.

Requirements/ goals	Grow the installed base as to obtain momentum (section 5) Manage flexibility and offer openness for evolution.
A set of system features	Evolving, shared, heterogeneous set of an installed base IT capability among a community of users (section 4.1.)
Kernel Theory	Complexity theory, evolutionary economics (section 5) • Enable organic growth and new combinations • Gain momentum • Recognize path dependency • Create lock-in through positive network externalities • Use modularity to offer organic growth and evolution
Design principles	A codification of five design principles which when applied will increase the likelihood of achieving a desired set of system features i.e. managed complexity, openness and growth in the installed base (section 6) • Design initially for usefulness • Draw upon existing installed bases • Expand installed base by persuasive tactics • Make if simple • Modularize by building separately key functions of each infrastructure, use layering, and gateways

Table 2. Components of a IS design theory for information infrastructures

Third, information infrastructures are at any moment of time *heterogeneous*: they contain components of multiple sorts – diverse technological components as well as multiple non-technological elements (individual, social, organizational, institutional etc.) that are necessary to sustain and operate the infrastructure. These components are connected in complex ways and they change constantly. This type of heterogeneity implies that infrastructures can and must be organized for future evolution through technical, institutional and social layering that enables controlled growth of heterogeneity. Therefore architectural control, architectural design principles and clean interfaces between layers of the architecture are critical not only in enabling heterogeneity, but also for amplifying it. Deployed architectural principles like standard families, related protocol stacks are critical in coordinating and controlling heterogeneity. IIs show multiple variations of heterogeneity as a result of the architectural layering: most IIs include sub-infrastructures at any point of time, which rely on different versions of some interface standard (e.g. different version of TCP/IP or different version of SAP); or different standards for the same IT capability (for instance different parts of the infrastructure running a different e-mail protocol).

Our II definition highlights two critical features that enable infrastructure design. Infrastructures must be *open* and they must be based on some type of shared *standards*. Openness in our definition signifies the lack of borders in an infrastructure in terms of its scale, scope or functions. This requirement is a function of the inherent need to support continuous growth and evolution. Infrastructures can not assume limits with regard to the number of elements they can include (applications being integrated, computers linked to the Internet, etc.), the number of users that will use them, or the number of functions or capabilities they can support.[3] Furthermore, an information infrastructure must also open in that it sets no principal limits, who can participate and contribute to its design. Lastly, openness is also temporal- an infrastructure has no clear start and end time - its development time is principally open.[4]

Standards "as general agreements between producers and users of technology" (David and Greenstein 1990) form another constitutive element of infrastructure design. Standards enable the evolution in scope and functionality, and they are a key means by which the infrastructure is architected and who is inscribed in its development. Standards offer means for organic growth of infrastructures in multiple ways. Infrastructures are first heterogeneous in the sense that they implement multiple versions of the same standard, or embed several standards for the same functionality. Two separate infrastructures can provide the same kind of services based on different protocols/standards but need to be linked together by standardized *gateways*. In this sense standard gateways increase infrastructure modularity and layering, and help *vertically* decompose infrastructures into separate *neighbouring* infrastructures.

II Standards can be either public or open in that they are either publicly agreed through some due process within specific institutional context (e.g. Internet or EDI

[3] Open as opposed to closed means that an infrastructure (or system or standard) is open with regard to who can participate in the design, implementation and use of it. This is many times of course violated in that architectural decisions close specific component from public control and exclude most participants from influencing this specific part of the infrastructure.l But the solutions often declared to be open, like the ISO OSI suite of protocols, is closed in the sense that it is designed under the assumptions that OSI protocols should only communicate with other implementations of the same protocols, and not allow (or support) gateways to other existing networks using different protocols.

[4] When arguing for the replacement of the concept of system with that of infrastructure, it would be more precise to say that the term system should be replaced by network to describe the change in our view of the inner structure of our ICT solutions. Then we should change our view on these solutions as seen from a user perspective: ICT solutions appear more as an infrastructure than as a tool. But we have found it more convenient to talk about infrastructures which also are open networks.

standards), or they can be adopted and enforced through market based mechanisms (e.g. Wintel standard set). Sometimes they can be institutionally enforced within a single organization. No matter which way the standards are adopted for II design their content and organization critically influences the future evolution of the infrastructure. In this sense we regard standards and infrastructures as two sides of the same coin. Standards describe the structure and functionality of the II while the infrastructure conforms to and implements the specified standard set.

Different IIs: A Preliminary Taxonomy

The general definition of II suggests a scope in which information infrastructures can vary enormously in terms of their scale and functionality: from more pervasive and generic, to limited and specific and more advanced (in terms of protocol layer). To distinguish between critical differences among different types of IIs the kernel theory needs to recognize between different kinds of information infrastructures. This taxonomy offers a basis for theory development and refined analysis of design dynamics of IIs related to specific design contexts. We will propose essential classifications of IIs by using the scale, scope and functionality of the infrastructure as the distinguishing factor. We observe two useful classifications: 1) one based on the vertical scope of the infrastructure, and 2) one based on the horizontal classification of basic II functions. All these infrastructures share the general four properties of infrastructures, but they differ in remarkably in their scale and capability.

Three types of vertical information infrastructures. Using the scale and scope of the II as the main classification criterion we can distinguish between three types of vertical IIs: 1) universal service infrastructure, 2) business sector infrastructure, and 3) and corporate information infrastructure. Table 2 clarifies four critical II features for each type of infrastructure.

Class of Infrastructure Feature	Universal Service Infrastructure (Internet)	Business sector Infrastructure	Corporate Infrastructure
Shared (by)	Potentially any application, service or user on earth.	Primarily companies within the sector (including their employees), but also customer and suppliers.	Primarily nits and employees within the corporation, but also suppliers, customers and partners.
Evolving	By adding services and computers to the network since the first packet switching network linking a couplet of computers were established	By exchanging new types of information among the users and by involving more organizations.	By integrating more applications with each other, by introducing new appli-cations
Heterogeneous	Many sub infrastructures, different version of standards, service providers, etc.	Multiplicity of competing and overlapping sub-infrastructures, standards, service providers, etc.	Multiplicity of applications and sub- infrastructures, users, services etc.
Installed base	The current Internet, applications integrated with it, users and use practices	All current integrated services, their users and developers, and the practices they are supporting and embedding.	All current applications and their users and developers, and the working practices they are supporting and embedding.

Table 3. Three classes of information infrastructures

The first type of infrastructure we call a "universal service infrastructure." Universal service infrastructure offers on the global scale transportation, access and storage services

using a set of open protocols that enable heterogeneous public connectivity to these services. The paradigm example of this infrastructure class is the Internet. Internet forms a universal shared information service infrastructure for all its hundreds of millions of users distributed across most countries of the world. It is widely believed to be **the basic future** infrastructure of the information society (Castells 19xx)[5]. Being a universal service infrastructure it at the same time offers the most important foundation - or infrastructure - to support the two other types of infrastructures. We will in this article mainly view Internet as a best practice to be learned from when theorizing about design of information infrastructures.

The second type of information infrastructure we call "business sector infrastructures." A typical example of this kind of infrastructure is EDI service networks for exchanging structured and formatted electronic documents between separate organizations (Damsgaard and Lyytinen 1999, 2001). Typical EDI documents include orders and invoices in transaction fulfilment cycles. More recently business sector infrastructures have grown into a broader variety of information service platforms that enable electronic markets and auctions, collaborative information sharing, business intelligence and the like within a business sector or among larger business communities. These growing infrastructures include various solutions for B2B e-commerce, telemedicine service networks, new types of web services (APIs) offered through Amazon.com, or Google and so on.

The third kind of information infrastructure we call "corporate information infrastructures." This kind of information infrastructures have emerged, when tel-ecommunication services offered distributed access to the internal information within an organization. Through this change, the number of potential users and uses that can be supported by internal information systems has grown exponentially. Furthermore, organizations integrate more their internal systems with those of their customers, suppliers, and strategic partners. This changes an organization's view and approach with regard to their internal systems. These systems are no more regarded as a collection of dedicated system functions, but rather seen as a complex web of IT solutions distributed across organizational borders, use areas, and user communities.

Three types of horizontal infrastructures. Another strategy in the design of infrastructures is to decompose a complex infrastructure into a set of simpler ones that offer only one type of functionality. This type of *horizontal* decomposition is equivalent to use of abstraction principles applied in software engineering (Parnas 1972). One type of layering is to split software functionality between business functionality (application) and infrastructure services (data base access, transportation and presentation layer). Using the layering principle we willl identify two types of horizontal IIs: *application infrastructures* and *support infrastructures*. These concepts are relative and will apply recursively. Any infrastructure can be split in its top layer to its application infrastructure and its support infrastructure upon which it is implemented. But the support infrastructure in turn can be further decomposed using the same criterion. We can further split any support infrastructure into two categories: *transport* and *service infrastructure*. These infrastructures are always dependent upon and related to one another through their service capabilities which are utilized when expanding and changing the infrastructure (see figure 1).

The transport infrastructure offers transportation services for other types of services. An example of a transport infrastructure is the basic TCP/IP transport infrastructure of the Internet that underlies all other Internet based services. Another example of a transport infrastructure is the SOAP protocol for exchanging XML based Web service messages between two web services. *Service* infrastructures provide necessary support for addressing,

[5] Naturally when most Internet users talk about using Internet they do not actually think about using the Internet service infrastructure per se but specific IT capabilities, which rely on it like WWW, IM, e-mail or blogs.

identification and service property discovery. A traditional example of a service infrastructure is the Domain Name Service (DNS) on the Internet, which is used virtually by all other Internet services to map textual identifiers like amazon.com (IP addresses, URL's, e-mail addresses, etc.) to numerical IP addresses. Other examples of services are UDDI and WSDL standards for Web services.

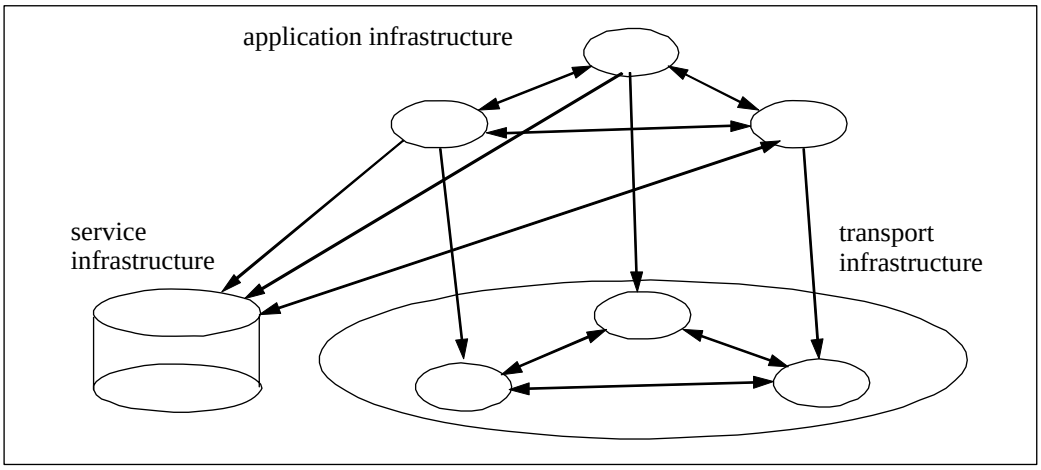

Figure 1. The structure of infrastructure

The Kernel Theory: How Do Infrastructures Evolve?

In this section we will solicit theoretical statements which govern the evolution and growth of IIs. We seek to address a question: how a design kernel theory can be formulated which help derive design guidelines for II growth and evolution. We will first discuss design requirements for IIs and then discuss elements of the kernel theory that would help create IIs that would conform to these requirements. We focus on two key properties of IIs: their complexity and heterogeneity, and the dynamics of installed base.

Design Requirements

When we normally characterize IS design goals they are stated in terms of fixed goals that can be achieved by manipulating specific instrumental variables in the design space (Simon 1981, Walls et al 1992). Examples of goals would be effectiveness, user satisfaction, system acceptance, system quality or maintainability. Due to the high complexity and low level of designers' control over goals of IIs we feel that II design goals must be stated differently. Specific design goals like user friendliness or cost-efficiency can be regarded as a contingent means to persuade or enrol specific actors to design or participate in the infrastructure (Callon 1986) at a specific point of time, or to mobilize bias to ensure actors' participation and justify their effort (Bergman et al 2002). Such goals, other than the general goal of growth do not apply, however, to information infrastructures as a whole. Dahlbom and Janlert (1996) characterize this difference crisply through their enlightened discussion of infrastructure *cultivation*:

> "[When we] engage in cultivation, we interfere with, support and control, a natural process. [When] we are doing construction (i.e. design in traditional sense OH & KL), [we are] selecting, putting together, and arranging, a number of objects to form a system.....

> [Cultivation means that] ...we .. have to rely on a process in the material: the tomatoes themselves must grow, just as the wound itself must heal." (ibid. p. 6-7)

This view is in perfect line with complexity theories (Cilliers 1996): infrastructures as complex systems evolve by themselves in unpredictable ways and therefore their designers need to set their goals differently. To consider technological systems as organisms with a life of their own implies that designers formulate their goals in terms of how they can influence the growth process through specific technological, social and political choices (in this view these choices are inseparable). Designers need to focus on how to enable the technology to evolve and remain open, and how this evolution promotes growth among the users and the technology. During II design it is thus as important to know and think what we have already and where and how the infrastructure as-is can "grow" in addition to what can be created in the form of a new artifact. This highlights the critical role of *existing technology, i.e. the installed base, as an actor* that affects design and which needs to be enrolled as an ally for design that seeks to grow the technology and its user base by coordinating design participants (non-technological actors) and design elements.

Design Kernel Theory

In order to understand how a designer can relate to an installed base as an ally and how he can cultivate the infrastructure into "independent" growth we will explore critical findings from the Social construction of technology literature (Latour 1991,1999, Hughes 1983, 1987), and evolutionary economics (David 1986, Grindley 1995, Shapiro and Varian 1999). Each of these streams suggests useful insights and laws how technological systems and their user bases can grow and sustain viability.

Explaining infrastructure growth. A turning point study in general understanding how complex technologies evolve was Thomas Hughes's (1983) eminent investigation of the electrification of Western societies during 19880-1930. He examined how the "networks of power" were erected into unprecedented large technical systems. His key insight is that at a certain moment such infrastructures obtain a *momentum* when the installed base becomes a new independent force affecting the future growth of the II. When this happens a self-reinforcing process of growth is set in motion and infrastructure expands fast "larger and more complex" (Hughes 1987, p. 108). This momentum marks a new type of independence in the technology evolution in the sense that the change associated with it becomes *irreversible*- the history cannot turned back and the system cannot be returned to its original state. In a way the genie is out of the bottle and there is no way the designers can put it back. Events, which can seriously threaten the momentum, are few and far between. Only conceivable in extraordinary instances can threaten it and will imply other significant negative consequences: "Only a historic event of large proportions could deflect or break the momentum [of the example he refers to], the Great Depression being a case in point" (Hughes 1987, 108).

But how does this momentum build up? Hughes' idea is that at certain points of time technical systems change their behavior and trajectory[6]. The key here is to explore the concept of irreversibility as it offers a means to understand how the dynamics of the complex system growth change with installed base. Irreversibility is also a key concept in complexity theory where it is formulated in terms of alternative state spaces for systems (or

[6] Hughes (1987) analysis is here more careful and insightful in particular his discussion of reverse salients and the necessity to overcome technological, social, political, economic and institutional barriers in building the infrastructure. But we will here analyze the economic demand side of the infrastructure growth which is not so obvious with electric power grids (which only benefit from economies of scale on the supply side).

attractor/threshold as it is called there). Irreversibility is explained by the concept of *path dependency* that complex systems exhibit in their time related behavior (Arthur 1994, David 1986, Cillers 1996). Path-dependency draws upon a cluster of other concepts in evolutionary economics including: increasing returns, positive feedback, network externalities, and lock-in (Shapiro and Varian 1999).

Increasing returns define demand-side economies of scale: the more a particular product is produced, sold, or used, the more valuable it becomes. In line with this IIs can become more valuable the more they have users. Most IT artefacts that form key components in IIs (telecommunication protocols, Data exchange protocols, operating systems, programming languages) have this characteristic (Arthur 1994). For example, the value of a compatibility standard's (e.g. a http communication protocol) is to a large extent determined by the number of users deploying it—that is, the number of users one can communicate with when one adopts the standard. In addition a large installed base attracts additional complementary products and makes the standard cumulatively (indirectly) more attractive (e.g. like other plug-ins that can be added to HTML documents; Shapiro and Varian 1999). A larger base also increases the credibility of the standard. Together these processes make a standard more attractive to potential users and thus expand the II related to that standard. This means more adoptions, which further increases the size of the installed base, and so on resulting in the positive feedback loop as a reinforcing mechanism (Grindley 1995 pp. 27). Increasing returns are results of (positive) *network externalities*, which arise when one participant affects others' value without additional compensation being paid (either positively or negatively).

Increasing returns in turn lead to another effect: *lock-in. A* lock-in means that, when a technology has been adopted, it will be very hard or impossible to develop competing technologies due to investments in the large installed base and resulting technological lock-ins. Thus positive feedback and resulting lock-in give rise to specific effects at the system level called *path dependence* (Arthur 1988). *Path dependency* suggests that specific past events (like adoption decisions, or correctly timed designs) have huge impacts for future design and the designs become path dependent due to lock-ins. In fact, events deemed irrelevant at their time can write history by having tremendous effects (David 1986). We can distinguish between two forms of path dependence: *cumulative adoption* and *technology traps*. The first one appears when a standard adoption builds up an installed base ahead of its competitors and becomes cumulatively attractive. In such a case the choice of the standard becomes 'path dependent' and is due to a small advantage gained during the early stages (Grindley 1995: 2). The classical and most widely known example of this phenomenon is the design of keyboard layouts that lead to *de facto* standardization of QWERTY (David 1986). The second – technology traps- relates to the fact that early design decisions will influence future design. When, for instance, a technology standard has been established and becomes widely accepted, new versions of this technology must be designed in a way that is compatible (in one way or another) with its installed base. Contingent design decisions made early on live often with the technology as long as it exists. Typical examples of this are IT technologies struggling with the backward compatibility like different generations of Intel's micro processors, where all later versions need to be compatible with the original 8086 processor.

***Explaining Infrastructure Flexibility*.** While the need to grow infrastructure and to create its momentum through standard adoption is widely accepted the need for infrastructure flexibility is less frequently admitted. We find that infrastructure flexibility is crucial for several reasons. First, as with al technologies first versions of any II are poor in quality. They are gradually improved while users get experience in using them and discover what is

needed both technologically and in terms of skills to use them effectively (von Hippel 1994, Orlikowski 19xx). During design time it is impossible to foresee all relevant issues, and many of them are discovered while users and designers go along within the technology path. For traditional ISs it is a well-known fact that their requirements will change over time because their environment -including user skills- changes (Lehman 1991). Likewise the successful growth of an infrastructure generates change needs as the II gains momentum. A good example is the "recent" redesign of IP addressing scheme due to the current version's (IPv4) limited addressing capability. When separate information infrastructures grow, additional needs emerge to integrate them into seemingly one infrastructure. The recent movement to integrate mobile (short) messaging and e-mail infrastructures or the Web and mobile portals (I-mode, WAP) are here cases in point.

We observe several types of flexibility that IIs need to possess in order to offer flexibility and to avoid technology lock-ins. The first one is that II needs to be easy to change: changing an II by replacing the current version of a standard with a better version should be possible and take place with low cost and high certainty. With IIs the goal is however, difficult to achieve as changes from one standard version to another add complexity and operational uncertainty- the infrastructures are brittle. A major difficulty that designers face is how to replace one standard version with another one, when the change will introduce backward incompatibility, which causes a technology lock-in. The second type of flexibility is use flexibility: an information infrastructure can be used in many different ways and serve different purposes. Use and change flexibility are related in the sense that increased use flexibility will decrease the need for change flexibility, and vice versa.

Explaining Infrastructure Growth. The challenges regarding design of II can now be formulated in relation two dilemmas that designers face when seeking to meet the II design goals. First, many if not most proposed infrastructure designs *never take off* (the dilemma of initial growth), as they never find a hospitable user community and their growth does not become self-reinforcing. As noted, infrastructures obtain their value from the relative size of their user community. They have initially no value. Accordingly, if no user finds early on an II useful its installed base never starts growing.

When infrastructures start to grow, the growth can become self-reinforcing when the infrastructure gains momentum. To succeed in designing such an infrastructure, one has to create i.e. "cultivate" conditions for such a self-reinforcing process to get started. If one succeeds, one can easily find oneself trapped in a *lock-in* situation both in terms of user base and technology capability. This is the second dilemma. There are two slightly different variations of these lock-in situations. First, designers can face a risk that separate users adopt different standards and thus incompatible infrastructures become established. In such a situation, it is often considered beneficial if all users agree and adopt one standard. But users might find the cost of switching too high when no compensation mechanism is offered. Thus a lock-in situation has appeared. This risk is often used as a justification for agreeing on one universal standard for an infrastructure before its development starts (De Moor 1993). If all users agree and adopt a shared standard, this standard will over time turn out to be inadequate when new technologies emerge. In this situation there will be another kind of lock-in - a technological lock-in. The design theory should help us observe such inappropriate situations and offer guidelines to avoid such situations or mitigate their impact whenever they emerge.

Design Principles for II Design Theory

General Structure of Design Strategies

This section will discern normative guidelines (procedural rationality), which need to be mobilized while designing IIs. Our main focus in formulating these design principles is to identify effective ways to manage the two design dilemmas. We need to find a way to formulate strategies that help build a new infrastructure[7] on the one hand -enable initial infrastructure growth-, and help change existing ones -support openness and evolution. These two design challenges are closely related. Changing an infrastructure means in some sense building a new one in a sense that new features must obtain their value from the size of the installed base. In spite of this, we will in the following distinguish between creating a new and changing an existing infrastructure. Making a new one means building an information infrastructure, which we do not expect to replace an existing IT capability. A case in point is current attempts to build e-commerce infrastructures like electronic markets[8].

When we seek to overcome these dilemmas we suggest one design strategy for each of them. Both of these strategies recognize the installed base (or its lack of) as the principal design target. The first strategy seeks to address the first dilemma by seeking to bootstrap and build up an installed base when there is none and to grow it. The second design strategy seeks to avoid the technology traps by making the infrastructure (and associated standards) as flexible as possible. Each of these strategies can be effectively executed by following five design principles: a) *For bootstrapping*: 1) Design initially for targeted usefulness, 2) Draw upon existing installed bases, 3) Expand installed base by persuasive tactics; b) and *for technology lock-ins*: 4) Make II simple, and 5) Modularize II by building separately key functions of each infrastructure, use layering, and gateways. Table 4 summarizes the content of each key strategy, the design principle and the nature of each design guideline. All these design principles are drawn from the kernel theory. Each principle in turn transformed into a number of design guidelines which are shown on the right side of the table 4. These design guidelines were formulated at the level where we can expect significant changes in designer's behaviors.

How to Bootstrap The II?

Design Initially for Usefulness. The first design principle suggests that initial users must be attracted for other reasons than the size of the installed base. Accordingly, a small (i.e. "manageable") group of potential first users need to be identified, and the first version of the infrastructure has to offer the group immediate benefits (guideline 1). This can for example be achieved by tailoring the first version of the infrastructure to serve the needs of the first users (guideline 2). First adopters accrue higher costs and face bigger risks. Accordingly, the version to be adopted by these users must be cheap and easy to learn in order to make the investments justifiable within a reasonably short time span (3). These users cannot be sure that the infrastructure will be implemented on a full scale or whether it will be in infrastructure at all. Accordingly significant investments may prove to be of less value because the lack of a large user base and associated benefits of demand and supply side

[7] Talking about designing a new infrastructure might appear in contradiction with my repeated claim that infrastructures are never built from scratch - only by enhancing and extending existing ones. This is not the case. The point is that any infrastructure requires a supporting one. The availability of such supporting infrastructures is an important success factor. The less modifications and extension of are required in existing infrastructures to make the new, the easier it is to build the new.

8 One might argue, however, that such electronic e-commerce infrastructures are replacing non-electronic infrastructures and that these also have to interoperate. Such integration will not be explicitly discussed here (Hanseth and Lundberg, 2001).

Key strategy	Design Principle	Element of Kernel Theory	Design Guideline
Bootstrap installed base [9]	**1. Design initially for usefulness** Design the II so that it is initially useful even though the fist users do not get any value from the number of users using the infrastructure.	Offer IT capability offered to and supported by a community	Target for a small group: Make it useful without installed base Make it simple to use and implement Design for one-to-many in contrast to all-to-all
	2. Draw upon existing installed bases Utilize existing infrastructures as much as possible in the diffusion of the infrastructure.	Use larger installed base as your ally Increase positive network externalities across communities Avoid dependency on infrastructures that are not available	Use existing transport infrastructures Design without the need for new support infrastructures Build gateways to existing infrastructures Use bandwagons
	3. Expand installed base by persuasive tactics to gain momentum Build an installed base as fast as possible.	Seek to increase positive network externalities Create lock-ins for users Offer additional value to users and expand learning in the user community as to enhance IT capabilities	Enhance the IT capability within the II only when needed Build and align incentives accordingly Develop support communities
Avoid technology lock-Ins	**4. Make it simple** Each element in the II should be as simple as possible.	Build system that enables community to grow and learn from their experience Use abstraction and gateways to simplify designs	Make it as simple as possible
	5. Modularize by building separately key functions of each infrastructure, use layering, and gateways	Account for unidentified needs Use means to separate concerns and simplify evolutionary decisions. Draw upon gateways and standards as to enable evolution at different parts separately	Divide infrastructure recursively into independent transportation, support and application infrastructures Use gateways between different standard versions Use gateways between different layers Build gateways between neighbouring infrastructures Develop transition strategies and support in parallel with gateways

Table 4. Design principles for a design theory of II

[9] For a more extensive discussion of bootstrapping networks and infrastructures see (Hanseth and Aanestad, 2003), and (Aanestad and Hanseth, 2002).

economies of scale[10]. Being cheap usually means that the IT capability is implemented with simple software and cheaper hardware. This gives further advantages: when the solution is small and simple, it will be easier and simpler to implement in user organizations thus lowering learning costs, and making it easy to integrate with existing infrastructure, Moreover, it will be easier to change when the infrastructure starts grow.

Each information infrastructure supports finally multiple information services. These can vary from services where "everybody" communicates with "everybody" (like e-mail) at one end to services where one user provides some information to all others. If possible, services supporting the information access or distribution from one point to a large group should be implemented first (4). This cuts down adoption costs and barriers and offers direct benefits for a group of key users.

Build on the Existing Installed Bases. The second principle suggests first selecting first supporting infrastructures that portions of the group of potential users use already (5). If an application infrastructure is designed so that it depends on a new support infrastructure, this will seriously build additional barriers to erect the new infrastructure. The required support infrastructure will be an infrastructure in its own right which needs to be bootstrapped first, or at the same time. This builds learning barriers and adds cost to the potential users (Attewell 1992).

Transport infrastructures are necessary to move the data between the users within any II and thus are the cornerstone for any II build up. In contrast, the value of service infrastructures is highly dependent on the size and sophistication of the installed based and its embedded IT capability. One does not need advanced service infrastructures when the installed base remains small. Such services become increasingly important when the infrastructure scales up. To render the infrastructure to be adopted by the first users easy to use and useful, one should therefore design it with the simplest possible service infrastructure (6). Service infrastructures should be implemented, expanded and enhanced incrementally whilst the infrastructure grows and the need for additional services becomes evident.

New service and application infrastructure should - whenever possible - be connected through gateways to neighbouring infrastructures. This gives the adopters additional benefits (positive network externalities) while communicating with the users of this infrastructure or accessing its information services (7). While linking the new infrastructure to existing ones, one should also take into account the speed and direction of their development and capitalize on bandwagon effects i.e. situations where users adopt fast its information services due to increasing positive network externalities (8).

Expand Installed Base Fast by Persuasive Tactics to Gain Momentum.The third design principle suggests that when users start to use a simple version of an infrastructure, its designer should try to find as many users as possible to adopt this version (9). This principle could also be described by a slogan "users before functionality." The phrase captures the basic idea of positive network externalities and positive feedback: an infrastructure will gain value primarily through the size of its user base, and not from the extensive quality of its functionality. New functionality should be integrated to the II only when it is needed i.e. when the scale the infrastructure obtains a new level, when it has so many users that its use value can justify the users to pay the extra costs of introducing additional functions, or when users learn to use it in new ways which enable to expand its functionality by learning by doing (9). This applies for all types of infrastructures:

[10] Here recent work real options on IT platforms can help. See e.g. Fichman (2004)

application, transport, and service sub-infrastructures, as well. Building up installed base demands to find ways to align divergent interests among different communities and persuade them to participate in using and designing the infrastructure. This will help enrol them as allies to promote future growth of the installed base (10). One critical element in this behaviour is to use the existing installed base as a platform for continued learning and support by creating and maintaining strong support communities (11) (Tuomi 2001).

How to Avoid technology Lock-Ins in II Design?

In the long term infrastructures need to be designed in ways that avoid technology lock-ins. In this regard it is obligatory to make infrastructures as flexible as possible – both in terms of use as well as change flexibility. We will here only look closer at the change flexibility as this feature is more important for the continued design of IIs. Use flexibility makes it less likely that the II will approach a technology lock-in but reaching such flexibility is extremely hard as it makes the original design of the II complex and its uptake difficult. Change flexibility makes it easier to get out of a lock-in when it appears which will appear anyway as the first designs normally cannot anticipate all types of uses and capabilities related to them. There are, in principle, two strategies to get out of a technology lock-ins: 1) an evolutionary strategy of maintaining backward compatibility or 2) a revolutionary strategy of sweeping transformation. These strategies reflect an underlying tension when the forces of technological innovation meet up with social world of network externalities: is it better to wipe the slate clean and come up with the best design possible (revolution) or to trade off performance to ensure compatibility with the installed base and thus ease user adoption (evolution) (Shapiro and Varian 1999)? As our primary strategy is to build incrementally a bigger II by linking it to the existing infrastructures we promote the evolutionary strategy. The alternative- the sweeping transformation through revolution- ignores the demand for backward compatibility and denies the use of the installed base. Because of this neglect it is inherently risky.

We propose two key principles that support evolutionary strategy in designing for flexibility with IIs. The first one calls for simplicity and leanness in basic architectural decisions concerning the II (12). Simplicity suggests that it is easier to change something small and simple than something large and complex as one will have to change the II anyway. The second principle recommends strong and clean decomposition of the II into several independent sub-infrastructures. Thus this decomposition must follow modularization and encapsulation principles of Parnas (1972). Each service must hide mechanisms that implement the service behavior at the interface and offer a loose coupling between services within sub-infrastructures. Any II should therefore be decomposed into several sub-infrastructures during design where each infrastructure offers “clean” interfaces to others so that they can evolve independently of each other (13). Infrastructures need to be accordingly decomposed horizontally into application and support infrastructures, and vertically into several independent neighbouring infrastructures. Designers should use gateways to connect together different regions of any infrastructure which run different versions of the same standard (14), or between different layers (15) or between related vertical infrastructures (16). Finally, transitions between incompatible standards need to be supported by appropriate transition strategies that take into account the impacts of change to installed base (17). The effectiveness of each transition strategy will depend on how each employs gateways, as well how the decomposition between horizontal or vertical infrastructures has been conducted.

Principle	Followed	NOT followed	Comment
Build up installed base fast			
1. Target the II for a small group	X		First used by researchers interested in accessing powerful computers. Later on e-mail was introduced to support just the operations and maintenance. The original web was designed to support collaboration among CERN researchers.
2. Design early on so that each service is useful without an installed base.	X	?	To our knowledge this was not an explicit design criterion. The developed technology was well suited for experimenting with distributed /Unix/ LAN technologies where global installed base was a minor issue.
3. Make it simple so that it is easy to develop and easy to learn to use	X		Obtain experience based on the use of simple prototypes and services. This has been an important design principle in the Internet community, in particular during the early adoption (Abbate 1999, Leiner et al. 1997)
4. One-to-many before all-to-all.	X		Remote login as a first service.
5. Use existing TIs.	X		A principal design principle in implementing the TCP/IP protocols was to make the Internet run on top of multiple underlying physical access layers: telephone, radio, satellite, LAN technologies, etc. (Kahn 198x).
6. Design without need for new SIs.	X		All early services were introduced without any new related SIs.
7. Build gateways to existing infrastructures	X		Gateways were developed to other e-mail protocols and services. Gateways to existing data bases and application have been important in making the web and web-services useful.
8. Jump on emerging bandwagons	X		The Internet technology was deployed during the 80's by joining its acceptance with the diffusion of workstation/Unix/LAN technologies.
9. Enhance the capability within the II only when needed	X		Many Internet services have been added when their needs have been recognized. The development of TCP/IP and its new versions IPv6, the introduction of DNS, enhancement of the web by XML and CCS are examples of this.
10. Build and align incentives to obtain allies	x		Internet offered low cost and innovative way for many scientific and engineering communities to communicate and share information and build associated services. Important allies were major research funding agencies and national research communities.
11. Develop support and learning communities	X		Development and use have been intertwined and the growth of associated support communities. Developers and users have both become important innovators for new services and organized support communities to do so. A large enough community (critical mass) was built originally to learn from distributed computing with workstation/Unix/LAN technologies. The similar applies to Web, Instant messaging or multicasting technologies. Currently large communities are built around process choreographies.
Avoid lock in through flexibility			
12. Make it simple so that it is easy to change	X		This guideline has been important throughout the whole history of the Internet and was explicitly stated early on (RFC 1994)
13. Split an infrastructure into independent sub-infrastructures	X		The key principle of "end-to-end" architecture promoted independence and modularization The Internet is actually composed of a large number of services, many sub-infrastructures are established and operated by a huge number of ISP's, etc.
14. Use gateways between old and new versions of the same standard.	X		The paradigm example of this is the use of tunnelling in the transition from version 4 to 6 of the IP protocol
15. Use gateways between different layers	X		This is key principle of the design of Internet which separate application, transport, addressing and physical connections. All these layers are connected through gateways.
16. Build gateways between neighbouring infrastructures	X		This was originally used e.g. to connect different networks (BITNET, Decnet) etc with Internet e-mail service protocols. The same took place with AOL and Progidy.
17. Develop transition strategies and their technological support in parallel	X		Support for the chosen transitions strategies was carefully introduced into the new version of the protocol itself.

Table 5. Internet design guidelines

Design of Internet as an II. We will next analyze how these design guidelines were mobilized in the design of Internet and how they influenced its success [11](table 5). This section will detail the evidence and narrate the application of each design principle during Internet design. The main thrust of our analysis is threefold. First, we use the Internet to illustrate concretely how these guidelines work and what their impacts when developing a universal service infrastructure.. Second, we want to demonstrate how these guidelines were derived from descriptions of the de facto (or emergent) design strategies that were followed during the design of Internet. These have been largely recorded in the operating principles of the governing bodies of Internet (e.g. IETF's RFCs in the Internet parlance).

Internet started originally as a small network aimed at providing a limited set of services to a limited and very select user group. It gained momentum after a decade and it has grown since then enormously. Through its history its designers, however, have been able to maintain it flexible and avoided harmful technological lock-ins - at least up till now.

Bootstrapping an Installed Base

Internet was Designed Initially for Usefulness for a Limited Group of Users. The Internet's success was largely due to the experimental, bottom-up and evolutionary bootstrapping oriented strategy (Abbate 1999, Leiner et al. 1997, Tuomi 2001). When the development of the packet switched technology started, those involved in the process had a great vision how the technology might be used in the future and how the future of telecommunications would be packet switching[12]. But their early focus was to provide a limited range of services from the start - remote login and file transfer – on top of the packet switching protocol that could stand a nuclear attack (1). The use was targeted to a rather small group of select researches, who were interested in accessing a few powerful computers (2). Among these, remote login was a perfect service to start with because each user could adopt it independently and they had the skills to do so (3, 4). And while the number of users using this service grew they could next start sharing data through file transfer. Later on, new services have been introduced following the same strategy (2). The e-mail service, for instance, was originally developed (by adapting a computer conferencing system running on a mainframe computer connected to the network) in order to support collaboration between the people responsible for the local nodes of the net at a time when only four computers were linked to it (Abbate 1999) (1). The web technology emerged through a remarkably parallel story. It was originally developed to support the collaboration among researchers attached to CERN (Berner-Lee 1999) (1).

The most important factor behind the successful diffusion - the real takeoff - of the Internet was the role this technology played within the research and development around distributed computing during the 80's (8). The primary focus among the distributed computing community was to develop an environment of work-stations and file servers connected by LANs. This development was organized primarily around the Berkeley Unix Version (BSD) of the Unix

[11] We do not claim that people followed these principles consciously. The choice may have been accidental, unconscious and dynamic and validation whether these were really consciously applied is very difficult. Our analysis reflects more like a rational reconstruction of some of the underlying design principles that emerged during the Internet evolution.

[12] The concept of packet switching was much older but it had been abandoned due to poor cost-efficiency and reliability and the technology lock-in for circuit switched networks took place during the late 19th and early 20th century (Tuomi 2001).

operating system (funded by DoD and DARPA) which became an important support and learning community (11). Its core component was the distributed system of file system, which again was based on TCP/ IP and other Internet technologies that came standard in BSD distribution files. The packet switching philosophy behind the Internet turned out to be perfectly suited for the vision of a future of computing consisting of a huge amount of interconnected LANs - a network of networks (which was opposite to the fundamental idea of traditional telecommunication design where there was just one network. Interestingly, this was an idea on which the Internet's rival, the ISO/OSI standard, was based upon (Abbate 199x)). As a result Internet technologies were adopted - more or less by accident - by many selected computer science research institutions and related broad engineering communities (software, EE etc) (10, 11). It was adopted as part of the workstation architecture used locally in contrast to enabling long distance communications. The Unix email system was originally ported to run on TCP/IP and later on increasingly sophisticated applications and services emerged (conferencing, remote games etc) that all run on top of the Internet protocols (10, 11).

The Internet itself was growing largely to support collaboration among the computer scientists developing this (distributed computing) technology (11). And the fact that the engineering community was running the same protocols within their organizations made it easy for them to link their computers together to support information exchange and collaboration across organizational borders.

Internet Built on Existing Installed Bases

The success of the Internet was highly dependent on exploiting available infrastructures as transport infrastructures (5). It was implemented on available data communication lines. For example simple modem based telecommunication links were used in some of the first versions of the Internet's e-mail service. The Internet's construction strategy was always experimental, and followed a bottom-up and evolutionary approach (3, 6). Each new layer or service (capability) has been established only at a time and when this service or layer can be made work satisfactorily (9). Further each layer has served as a platform for developing new and more advanced or specific services (Abbate 1999) (5, 7, 9). Currently, Internet provides critical services for electronic commerce including support for transactions, identification and security that have been specified and built as yet another layer on top of the existing installed base like TCP/IP and http (see e.g. Nickerson and Zur Muehle 2003, Gosain 2003, Faraj et al 2004).

Currently Internet contains multiple services, which were not part of its initial specification (6). An illustrative one is the specification and evolution of Domain Name Service (DNS). It is a global service infrastructure on its own on Internet that runs on top of Internet's TCP/IP[13] based transportation infrastructure and offers a service which maps symbolic domain names to physical IP addresses. During the early days of Internet service the need for such a service was not obvious. If it had been part of the original design, however, it would have made the building and diffusion of Internet more difficult. During the early development of Internet, the novelty of the packet switching technology made the whole undertaking complex and risky. Adding more functions would have resulted in more complex design and implementation risks. This would have made an early failure of the technology more likely. The need for additional support infrastructures like DNS was discovered when the Internet started to grow. Yet, the scale of the system was still relatively small so that it was easy to design and implement a name

[13] DNS is in fact using UDP and not TCP. The same is also true for the ping, talk, and finger services. UDP is a simpler and less secure alternative to TCP.

service and link it to the stable and well functioning transportation infrastructure. Later, DNS has become crucial in helping the Internet to scale up and promoted its later success by promoting ease of use and effective naming of services and sites[14].

The Internet has during its evolution increased its installed base by building gateways to existing infrastructures (7). Here the web is a case in point. The web became increasingly useful (and popular) when HTML based files could be obtained from web servers in the Internet using the http protocol. A huge part of this increased value was created through leveraging upon existing data in enterprise wide databases and applications (dynamic web or "deep" web). Data from these systems were fed into web servers through gateways that could execute scripts (embedded in html) using Common Gateway Interface standards (cgi).

Internet Built an Installed Base Fast

The experimental approach that guided Internet development guaranteed that simple versions were always designed and implemented first (12). The focus in this way was always turned towards diffusing existing services as-is as to foster community based learning (3). This strategy was at the centre of the controversy between the Internet community and the ISO/OSI effort concerning a suitable standardization approach. The OSI community argued that Internet protocols were lacked important functions, and that these should be added before they were implemented and diffused. In contrast, the Internet community advocated a grass-roots strategy that emphasized simplicity and running implementations. New functions should be only added when deemed necessary (Hanseth et al. 1996, Hanseth 2002) (17) and Internet has continually grown when new functions have been added to it. DNS service offers here a paradigm example.

Internet Offered Adequate Flexibility

Over time Internet has proved to be remarkably flexible. Its protocols are lean and simple (in particular when compared its OSI counterparts) (12). Thereby design ambiguities and errors were more easily avoided during the implementation process, the resulting technology is leaner and simpler and accordingly easier to change. When the first version was tried out it generated new needs and requirements. A critical consequence of the experimental strategy was that the Internet's development confronted at an early stage challenges related to information infrastructure change. This need to prepare for continued change is expressed eloquently in a document describing Internet's standardization process: "From its conception, the Internet has been, and is expected to remain, an evolving system whose participants regularly factor new requirements and technology into its design and implementation" (RFC 1994, 6). Accordingly, design challenges for flexibility were early on taken into account while adding new features as well as in formulating associated design strategies. The central gist of the resulting design approach has been nicely expressed in the slogan "we believe in rough consensus and running code." The anticipation of change was also reflected in information sharing and learning: a special report on the evolution of designs was issued quarterly to keep track of all the design changes and to confront them with criticism (RFC 1995). Further, the rules of the Internet standardization dictated protocols how to advance a design idea from a proposed draft into a new full Internet.

One important reason for the speed and scope of innovation within Internet was its initial modular design (13). Future flexibility was a key issue from the beginning in designing packet

[14] The name server itself has gone also though some changes like the adding of means to increase the address space (e.g. NAT), or organizing the name service for a newly designing addressing scheme (IPv6)

switching technologies. Internet's end-to-end architecture, which demanded putting the "intelligence" into the end nodes and not into the network as in traditional telecommunication networks, proved to be especially significant architectural choice, which enabled a very different trajectory for the continued growth of the infrastructure (14, 15, 16) (David, Lessig, Abbate). This architecture has stimulated local and largely independent development of new services within the community (See e.g. Tuomi 2001, Rheingold 1992) and resulted in multiple independent service capabilities that were laid on top of TCP/IP transportation infrastructure like MIME (for e-mail), ftp (for file transfer), MBONE (for multimedia streaming), or WWW (for information sharing) (15). Each of these standard sets could be again maintained independently of the underlying transportation infrastructure.

Gateways have played, and still do, important roles in the evolution of the Internet. Multiple gateways prevail, for instance, between the Internet's e-mail service (MIME) and numerous other networks that run proprietary e-mail protocols (16). Another important family of gateways is those found between the Internet information access services and internal applications and data bases connected to the "net" through web servers that communicate through Common Gateway Interfaces (cgi) (16).

Avoiding confrontation between different alternatives in infrastructure design was particularly important during the early design of Internet while considerable uncertainty reigned with regard to how the infrastructure would evolve (14). This uncertainty could not be settled up front due to the lack of use experience. It can only be resolved gradually. The use of gateways prevented designers to act like 'blind giants' by making early decisions easier to reverse. This role of gateways is nicely demonstrated by the role played by a specific gateway, called the Nordunet Plug, in the early adoption of the Internet among Nordic countries (Hanseth 2002) in the mid 80's (15, 16). The project aimed to develop a shared network among all Nordic universities. In the start all participants arrived easily at an agreement of the desirability of such a network. But the project remained deadlocked for a while because the participants could not agree which protocol should be used in for the transportation service. The deadlock was resolved through an implementation of a gateway that offered a set of shared services between all existing networks (in contrast to building a new one) (16). The gateway made it easy, by accident, for users to move independently to the Internet services, which surprisingly happened rather fast and without any conflict.

In the past several Internet protocols have been upgraded (14, 17). An interesting example of how flexibility has been maintained is the design and implementation of IPv6. The case illustrates well the need for II change in light of scaling up and new technological capabilities as well attempts to find alternative transition strategies for overcoming installed base inertia (Hanseth et al. 1996, RFC 1994, Monteiro 1998). During the period between 1974 and 1978 four versions of the bottom-most layer of the Internet, i.e. the IP protocol, were developed and tested (Kahn 1994). For almost 15 years IPv4 was practically very stable and enabled and supported the fast growth since the late 80's when Internet gained momentum. When people noted that the address space for internet was running out and the IP addressing scheme did not support well new data communication requirements (like multicasting, mobility of services within the network) the work on a new version of IP started in the early 90's. A detailed list of requirements was worked out, and several protocol alternatives were proposed. The final agreement was reached concerning a new IP, called IPng (new generation), or IP version 6, in 1994. The accepted version, however, fulfilled just a few of the requirements that had been originally listed. The most important satisfied requirement is a new address scheme, which

significantly extends the address space. The most important design criterion turned out to be- not what the new addressing scheme would be- but what specific solutions are needed for a stepwise introduction of the new version into the installed base (17). Initially this criterion was hardly mentioned in the requirement lists. But while designers inquired deeper into the critical design constraints, the more important it turned out to be. Towards the end of the standardization process this transition strategy was considered to be the most important design issue together with the rationale of the extension of the address scheme. This specific criterion, of course, constrained what possible solutions could be proposed for other registered needs (RFC 1995). At the end a majority of the other requirements were dropped from the standard, because the designers did not find efficient ways to fulfil them while version 4 compatibility was maintained for the new standard. The large number of users and user organizations now involved in the standardization process also increased the difficulties in reaching agreements (Steinberg 1995). It was expected that the transition from the old to the new version would take years or even a decade. After a decade of formally agreeing on then standard, the first commercial implementations of the new protocol have just become available, and the transition is still in the early stages (van Best 2001).[15]

The Failure of II Design: EDI in Norwegian Health Care

Next, we describe a failed effort to design an II for healthcare in Norway. The main findings of this analysis are portrayed in table 6. The motivation of narrating this story is also twofold. First, we want to illustrate how *not* following the guidelines in designing a business infrastructure can lead to a failure. This demonstrates that the principles stand the falsification test. This generalizes the validity of these principles across different families of infrastructures. Second, the case shows how some of the guidelines emerged from investigating the meager success of this effort.

The development of EDI infrastructures for health care in Norway has followed a design strategy, which is different from the one followed with Internet. The approach resembles closely the traditional specification driven software development approach- or the life cycle design theory. This has been conventionally followed in many standards developments in several areas including telecommunications (Schmidt and Werle 1998). The same approach was adopted by the ISO/OSI designers, who fought a "religious war" with the Internet developers for many years - but in the end lost the battle bitterly (Drake 1993, Schmidt and Werle 1998). Many scholars have interpreted the demise of the OSI standard set to be due to its disregard of the evolutionary approach (Stefferud 1992, 1994; Rose 1992). We concur with this view. Yet, we also believe that the same reasons can account the sparse positive outcomes of developing information infrastructures for health care.

Building the Installed Base

An information infrastructure for health care sector in Norway started when a private laboratory developed technological solution enabling electronic transmission of reports from the

15 This illustrates that the Internet is not at all changing as fast as the hype tries to make you believe. For more on this issue, see (Hannemyr, 2003).

Design strategy: Build installed base			
Design guideline	Followed	NOT followed	Comment
1. Target the II for a small group		X	In the first phase each laboratory focused on the general physicians as main (potential) customers. These client groups are not truly small but manageable. Later on the focus shifted on enrolling all potential actors within health care.
2. Design so that is useful without an installed base		X	No specific focus was placed in finding beneficial services early on to any of the participant groups.
3. Make it simple so that it is easy to develop and easy to learn to use	(X)	X	Early laboratory report services were simple. In the second phase the early services were dropped and the specified solutions were complex.
4. One-to-many before all-to-all.	X		Services distributing laboratory reports were exactly of the one-to-many kind, but were dropped in favour of all-to-all services.
5. Use existing TIs.	(X)	X	The laboratory report services relied on existing TIs (terminal emulators based on telephone services). In the second phase, a brand new TIs was planned and assumed to be established.
6. Design without need for new SIs.	(X)		The laboratory report service included no new SIs. In the second phase, however, several new SI's were introduced.
7. Use gateways to existing installed bases.		X	The second phase did not draw upon existing installed base as a beachhead for expanded adoption. No specific gateways were designed or considered for any of the existing systems among different stakeholders (pharmacies, hospitals, physicians)
8. Jump on bandwagons)	(X)	X	Many actors involved in the second phase believed that the future infrastructure would be based on the OSI protocol. In this sense they also believed they had jumped on the right bandwagon which was not the case as no careful analysis was carried out how OSI could become a bandwagon when it in fact was not.
9. Installed base before functionality. Enhance the capability within the II only when needed.		X	The strategy was to specify standards in abstract, and the infrastructure would automatically implement itself. No consideration was given on the existing installed base.
10. Build and align incentives to obtain allies		x	The strategy was not followed as designers did not explore incentives by which different communities and actors could be enrolled to using the design infrastructure ad/or which ones were critical to support it.
11. Develop support and learning communities		x	There were no attempts to develop garner or distribute learning experiences or create separate support communities related to the design, use or implementation of planned services.
Design strategy: Avoid lock-in through flexibility			
Design guideline	Followed	NOT followed	Comment.
12. Make it simple so that it is easy to change		X	The design was the opposite: Design the infrastructure and its standards so that all future requirements are satisfied and future changes can be avoided
13. Split an infrastructure into inde-pendent sub-infrastructures	X	X	This guideline was followed in the sense that the overall infrastructure was split into layers. Yet, it was not split horizontally into different national standards, nor specific industry sector standards (hospitals, physicians, pharmacies)
14. Use gateways between different versions of the standard.		X	Issues related to the evolution of standards were neglected.
15. Use gateways between different layers		x	Not followed as new transportation infrastructure was expected
16. Build gateways between neighbouring infrastructures		x	Not followed. The design approach did not use the installed base to integrate with neighbouring infrastructures.
17. Develop transition strategies and their support in parallel		X	Issues related to the evolution of standards were neglected.

Table 6. Design of EDI infrastructure in health care

laboratory to General Physicians (GPs) (1). The laboratory developed the solution because they wanted to attract more customers. They believed that GPs would appreciate receiving these reports electronically so that they could be stored into their patients' electronic medical records. The solution was simple. It was implemented by using a terminal emulator software package operating over telephone lines. The laboratory gave it to GPs for free together with modems (3). This solution was highly appreciated by the many of the existing customers" and it attracted many new ones (2). The success of this solution caused many other laboratories to imitate it.

The positive experience with these laboratory report services made several actors in health care (doctors, health care authorities, software vendors, etc.) to realize that information services could be successfully utilized within a broad range of inter-organizational transactions. This included transfer of multiple types documents that had much in common with laboratory reports: laboratory orders, admission and discharge letters, reports to social security, prescriptions, etc. Each of these information exchanges can involve several institutions. Accordingly, exchanging all this information electronically would require erecting a shared and open information infrastructure for the health care sector.

When exploring the design alternatives for such an infrastructure, the participants discovered that many data elements were included in several information exchanges. Multiple constraints and interdependencies were detected among these data elements. A consensus was reached that the best way to design the II was to develop a family of solutions that would use the same reference model that would define all data elements, their organization and use in information exchanges. Tin short, the sector would need to define a coherent and complete set of information exchange standards that would rest on an industry wide information model. Single transactions would be defined as "views" of this model. The model should be developed by following an IS design method where designers derive user requirements by modelling the "real world" (Davis 1982).

It was soon also observed that in the best situation these information exchange standards could be global. Patients increasingly travel all over the world, and the health care sector increasingly buys and offers health care products and services globally so benefits of a common data definition standard would be significant (De Moor 1993). Therefore, since the early 1990's the focus shifted in formulating a shared set of European health care standards (CEN TC/251).[16] In this process it was assumed that when these standards are defined, they would diffuse automatically and be implemented through a "big bang" approach. Hence the approach did not seek to draw upon and grow installed base first violating the first four design guidelines (-1,-2, -3, -4) [17]. The standardization design approach assumed also that there was no need for experimentation and building on observed needs (-8, -9).

This "design from scratch" is well illustrated also by the fact that the standardization body decided that all information exchanges need to be transported through a new underlying (transport) infrastructure that was based on ISO/OSI X.400 protocol[18] (-5). This resulted in a situation that when the first information exchange standards of CEN TC/251 had been specified, there were no available X.400 based services to transport the specified message. The need to build a complex X.400 transportation infrastructure increased also later implementation problems. In spite of broad political backing of the choice (ISO/OSI model was backed by EU and thus had high level political support), user organizations were reluctant. The reluctance was

[16] CEN is the European branch of the International Standardization Organization (ISO).
[17] The notation –x shows that the specific guideline was not followed.
[18] X.400 is the ISO/ OSI standard protocol for e-mail.

due to X.400 implementation complexity and its lack of installed base. Most organizations had other e-mail systems which interoperated with most other systems through a patchwork of gateways while there were no gateways to X400 services in most systems (-7). Organizations had difficulty in accepting that they could not use existing messaging services and instead install another e-mail system. This lead to a "penguin effect"[19]: everybody was watching all the others and waiting to see whether the others would install X.400 (-9).

EDI information infrastructures need multiple service infrastructures on top of the message based transportation infrastructure (-6). This in particular, covers multiple identification capabilities and associated naming infrastructures. Some these have been designed as part of the ongoing standardization work (like e.g. EDI user directories), while others have emerged through more general efforts to build social infrastructures for health care (Bowker and Star 1999). These include among others disease, drug classifications and registers for licensed physicians. For example laboratory report must identify associated test order, the ordering unit (GP, hospital department, another lab, etc.), the patient; prescription messages must identify the prescribed drug, the patient and the ordering unit (doctor) etc. Registers of such identifiers must be available to all information infrastructure users and they must be common to avoid mistakes and errors. This requires building an additional information infrastructure to maintain and distribute updated versions of each register and a social and institutional process to define classification semantics (Bowker and Star 1999). Such information infrastructures can become rather complex technologically but even more so organizationally and politically. At the European level, the political and institutional process of formulating a standardized Pan European system of codes to identify the specimen analyzed (blood, skin, urine, etc.), where on the body it is taken, the types of tests to be performed, and how their results are represented and communicated has so far turned out to be slow and cumbersome, and so far has failed. The problem faced is partly to find an arrangement that would serve the needs of all countries and different stakeholders. But more difficult has been to find a solution which can easily replace the installed base of multiple national and institutional systems at a reasonable cost and risk (-6, -7).

The design of a drug identifier system was one necessary service infrastructure when a general solution to transfer electronically prescriptions from GPs to pharmacies was developed (Hanseth and Monteiro 1997). The solution agreed upon during the design was to piggy back on the installed base of pharmacies' inventory systems (7) and use the same identifier codes in transmitted prescriptions as those used by pharmacies use in their internal inventory control systems. This solution required to design an additional infrastructure to generate updated versions of these identifier lists for GPs, to distribute these versions electronically to the GPs, and to install new versions of these identifier lists as a part of the electronic medical records in hospitals. When the implementation of this system started it was a big surprise for the designers that it was beyond their reach to establish such an infrastructure as they had neither institutional control nor political power to influence GPs or hospital systems (-10, -11). Establishment of registers and related infrastructure services reach often this level of institutional and political complexity when incentives are not there and interests aligned, and therefore fail. To make the system work locally the first version of the infrastructure for electronic prescriptions identifies the drug by its name as in the current manual version (9). Until the drug naming infrastructure is

[19] This is a situation where a flock of penguins wait on a cliff which one of the penguins will jump first into the water as a way to test if there are any predators in the water.

established and adopted extensively, this could be a perfectly adequate solution.20.The de-facto, emergent strategy followed in developing the "first generation" laboratory report solutions was successful. It looked like a perfect start to build an information infrastructure and grow its installed base21. The design strategy adopted in the next phase, however, differed and a closed specification driven approach was adopted largely influenced by the ongoing ISO/OSI standardization effort. Since the early 1990's significant efforts have been poured on standardizing health care related information services. A large number of more or less complex standards have been specified. Some of them have been even implemented in small and isolated service networks and communities. But, by and large, no general infrastructure has been built. The lack of progress after more than a decade of standardization can be explained by the complexity of working out complete specifications of the information to be exchanged which satisfy the needs for the whole health care sector in Norway and later on in Europe. But the major reason for the failure is ignoring the complexity and cost involved in implementing "complete" standard specifications that ignore specific dependencies in the design environment that help bootstrap the installed base. The strategy that was followed in building an installed base followed exactly the opposite guidelines than what the rules 1 to 9 recommend. Further, no attention was paid that these standards would change anyway after the implementation and thus also design guidelines that would foster flexibility were not enacted [22].

Concluding Remarks

We have argued in this paper that many contemporary IT solutions reach complexity and longevity that goes beyond what can be coped by using traditional design methods. Accordingly we need to theorize in a new way how to design such infrastructural IT solutions. To this end we have formulated in this paper fragments of a design theory of information infrastructures. We have defined its key unit of analysis and its critical properties, the kernel theory of how infrastructures evolve and grow, and formulated a set of design strategies for II design. The new design theory focuses on the interactions of the context of design and the goals of design where installed base forms a critical design factor. Accordingly, design of IIs is metaphorically viewed installed base cultivation i.e. creating conditions for the II to evolve and grow.

In this section we advance some arguments about the status and nature of the proposed theory: its completeness, validity and practical implications. All theories, including the one we have put forward, are by design incomplete (Weick 1986). In this context we will address this incompleteness in two respects: 1) not all issues are covered in its current formulation that is relevant in the assumed scope of the theory, 2) its external validity and applicability.

Though our theory draws upon evolutionary economics the current level of theory development does not offer detailed normative ways to make design decisions based on economic calculus. For example, the theory does not provide much help in trying to estimate the

[20] Another example of a service infrastructure for health care infrastructures is the ICD system for classification and coding of diseases whose infrastructural character is discussed at tremendous depth by Geof Bowker and Leigh Star (1999). Several discussions of registers of this kind can be found in (Bud-Friedman 1994).

[21] Nobody really seemed, however, to pay attention to the relationship between the solutions developed first and the more general infrastructures that could be developed in the future though.

[22] This is not true completely in that new versions of EDIFACT messages were taken into account by giving each version a unique version id. But beyond this trivial decision, this issue was overlooked.

economic benefits of making a specific infrastructural choice[23]. In its current form the theory is primarily a constraining design theory: it states what one should not do, rather than advocating what one should exactly to in a given situation. Thus we regard design constraints more important than specific design goals for a design theory. The theory is also limited within this scope. For instance, it does not say anything about the politics of infrastructure development and how to identify, or cope with the power as a design constraint. To do so we need to expand the current theory with other theories like actor network theory (Latour 1991, 1999), or institutional theory (Scott 2001).

The theory is also limited in that we strove for higher level of generality instead of accuracy while maintaining the theory relatively simple. The theory is currently composed of a few abstract concepts and it cannot account for all pertinent II design phenomena at a detailed level. Suggested concepts are general abstractions that are common to many complex IT application domains. By making this choice we purposefully left out significant differences among infrastructures. The power (and usefulness) of our theory derives from its simplicity. But the simplicity comes at cost: the strategy guidelines offer no "silver bullet" for specific situations. We can certainly find cases where specific contingencies may demand more detailed strategies. At the moment our theory has at best pragmatic legitimacy (Robey 2004) in that its predictions can lead to useful behavioural guidelines instead of suggesting the "only" or the "best" solution.

In order to advance the theory to a more sound design theory it needs to be more vigorously validated. In this paper we validated the theory's predictions through two case studies that ranged over different contexts (theoretical sampling). We validated the theory both through corroboration (how the design theory lead to design guidelines that lead to achieving design goals) and falsification (how *not following* the guidelines derived from a theory lead to not successful design outcomes). A critical counter-factual question to evaluate the validity of the theory is also the following question: had it made a difference if designers had pursued a strategy based on the design theory for IIs in the second case ?[24] We cannot be completely sure about this, as we cannot execute another design experiment followed a different strategy under the same conditions. Based on our analysis of the case we still however argue that if designers had followed another design strategy, better results would have resulted. This strategy would have started with the experiences gained from the first laboratory report services and proceeded as follows:

1. Improve and extend the existing infrastructure for laboratory reports; (2, 9, 5)
2. Harmonize the used message formats into more general standards (6, 9, 10, 11).
3. Link the different transport infrastructures together through a gateway strategy. (7, 8)
4. Build similar infrastructures for other promising areas (other kinds of laboratory reports, then to other kinds of forms) (1, 2, 3, 4)
5. Improve existing solutions based on experience how the IT capability shared across different communities enable better and/or more efficient health care service (9, 10, 11).

[23] Here real option analysis (Fichman 2004) can be useful and the theory could be integrated with a more analysis of real options that follow from different infrastructural choices.

[24] Following falsification principle followed here this knowledge is always contingent in the sense that we can never be sure that the strategy would work successfully in all future cases i.e. the theory is falsifiable.

We believe many problems would have been solved by focusing on developing national infrastructures for specific constituencies and linking them together. The laboratory pioneering transmission of reports in Norway, for instance, had already set up gateways to several non-Norwegian pharmaceuticals (that received the laboratory reports related to testing of new drugs) (Hanseth and Monteiro 1997).

Due to the space limitations and research choices we have not been able to exclude other explanations for observed outcomes. We did that during our research process but could not find better explanations for observed outcomes. Moreover, the search of alternative explanations in cases like this is not fruitful as our primary concern is to illustrate the power of the proposed theory and first show its validity.

We have neither examined cases where not following design theory guidelines lead to successful outcome, or where following these guidelines lead to failures. We are quite confident that we can find instances of both theory/outcome combinations which shows probabilistic (pragmatic) nature of the proposed theory, and the omission of important elements that will affect design outcomes (like power, lack of detail in discussing architectural principles)[25].

In future the proposed theory needs to be validated by being consciously applied for infrastructure design. Such validation is, however, no simple task. The specific "design" approach suggested by the theory needs to be integrated with everything else that is going on during infrastructure development over extended period of times. All this makes controlled validation of the theory unrealistic. As we have argued the fundamental principles that underlie the theory are solid and widely accepted. Our contribution has been to formulate a platform how to use them effectively design complex IT artefacts. In future research we plan to validate the design theory can be by using it to predict outcomes of a selective set of infrastructure designs that cover a broader range of types of infrastructures, and alternative design strategies. Interesting cases would be the current design of web services (Nickerson and Zur Muehlen 2003, Gosain 2003), the design of infrastructures based on new vertical industry standards like design of broadband mobile services in different contexts (Yoo et al 2004, Tilson and Lyytinen 2004).

We see great practical needs for (improved) II design theories due to the increased infrastructural nature of IT investment (Carr 2003) while its scale and scope (Weill and Broadbent 1998) extends. Accordingly, we also see significant practical and managerial implications that flow from our theory. If the theory was adopted, the practice of designing infrastructures would change significantly. This change would imply a new way of thinking of design elements like simplicity, openness or architectural control. Making the change real is - at least - as challenging as it is radical. Industrial practices are complex interconnected heterogeneous networks. And, a "paradigm "change within such a practice would imply the abandonment of the most fundamental thought "standards" with incompatible ones. Such revolutions happen rarely. Industries and their designers are usually locked into thought patterns that are not dropped when they were falsified and the designers continue to learn superstitiously (March 1991). A paradigm change of this scale requires more research on all aspects of infra-structures: 1) what contributes to their complexity, 2) how the complexity attributes them a sort of agency which is out of designers' control, and 3) and how infrastructures grow and decline under such circumstances.

[25] The theory is so intimately linked to network economics, that falsifying these principles under all circumstances would be close to falsifying established theories of network economics. This theory has a solid basis, although all its predictions concerning market failures are not unanimously supported (see e.g. Margolis and Liebowitz 1995).

Standards form the core of any infrastructure. Accordingly, much of this research need to focus on standards: their design, diffusion, evolution, and impact. At the same time we are developing new *kinds* of standards as IT becomes pervasive and key coordinating element in industrial organization (Markus et al 2003). For example, specific working practices have become deeply embedded into EDI standards in health care (Hanseth and Monteiro, 1997), or new Web service standards in financial services (Markus et al 2003). When such standards become widely adopted, their inertia may have signification implications of how working practices may and will evolve and their mindless design can plant the seeds of organizational inertia and behavioural lock-in.

References

Aanestad, M., and Hanseth, O. (2002) Growing Networks: Detours, Stunts and Spillovers. In Proceedings from COOP 2002, Fifth International Conference on the Design of Cooperative Systems, St. Raphael, France, 4.-6th June 2002.

Abbate, J. (1999). *Inventing the Internet*. MIT Press, Cambridge, Ma.

Arrow, K. J. (1994) Foreword. In (Arthur 1994)

Arthur, W. B. (1994) *Increasing returns and Path Dependence in the Economy*. Ann Arbor: The University of Michigan Press.

Arthur, W. B. (1988), Competing Technologies: an Overview. In *Technical Change and Economic Theory*, Ed.: Giovanni Dosi et al., p. 590 - 607, Pinter Publishers, New York.

van Best, J.-P. (2001). IPv6 Standardization issues. In K. Dittrich and T. M. Egyedi (eds.) (2001). *Standards, Compatibility and Infrastructure Development. Proceedings of the 6th EURAS Workshop,* Delft University of Technology, Faculty of Technology, Policy, and Management, Delft, the Netherlands.

Bowker, G. and Star, S.L. (1999). *Sorting Things Out. Classification and its Consequences.* MIT Press, Cambridge, Ma., 1999.

Brooks. There is no Silver Bullet.

Bud-Friedman, L. (1994) (ed.) *Information acumen. The understanding and use of knowledge in modern business*, pages 187–213. Routledge.

Callon, M. Techno–economic networks and irreversibility. In Law J., editor, *A sociology of monsters. Essays on power, technology and domination*, pages 132–161. Routledge, 1991.

CEN TC251/PT03-025.(1995). Methods for the development of Healthcare messages. CEN, Brussels.

CEN TC251. (1996) Standards in Health Care and Telematics. CEN, Brussels.

Cerami, E. (2002) *Web Services Essential.* O'Reilly, Sebastopol, CA.

Ciborra, C. U. Introduction: What Does Groupware Mean for the Organizations Hosting It? In C. Ciborra (Ed.), *Groupware and Teamwork.* New York: John Wiley & Sons. pages 1 - 19, 1996.

Cilliers, P. (1998). *Complexity & Postmodernism. Understanding Complex Systems.* Routledge, London, UK, 1998.

Dahlbom, B. and Janlert, L. E. (1996) *Computer Future*. Unpublished manuscript.

David, P.A., (1986) Understanding the Economics of QWERTY. In *Economic History and the Modern Economist,* edited by W. N. Parker. Basil Blackwell.

David, P.A., and Bunn, J.A. (1988), The Economics of Gateway Technologies and Network Evolution. *Information Economics and Policy*, vol. 3, pp. 165-202.

De Moor, G.D.E. (1993) Standardisation in health Care Informatics and Telematics in Europe: CEN TC 251 activities. In (De Moor et al. 1993).

De Moor, G.D.E., McDonald, C.J., and Noothoven van Goor, J.(eds.). (1993) *Progress in Standardization in Health Care Informatics.* IOS Press, Amsterdam.

Drake, W.J. (1993). The Internet religious war. *Telecommunications Policy*, Vol. 17 No. 9.

Fichman R. (2004): "Real options and IT platform adoption: Implications for Theory and Practice, *Information Systems Research*, 15,2, pp 132-154

Fitzgerald, B. (2000). System development methodologies: the problem of tenses. *Information, Technology & Peopl*e, Vol. 13, No. 3, pp. 17-185.

Grindley, P. (1995) *Standards, Strategy, and Politics. Cases and Stories.* Oxford University Press, New York.

Hagel III, J. (2002) *Out of the Box. Strategies for Achieving Profits Today and Growth Tomorrow through Web Services.* Harvard Business School Press, Boston, MA.

Hannemyr, G. (2003).The Internet as Hyperbole. A Critical Examination of Adoption Rates, *The Information Society.*

Hanseth, O. (2001) Gateways - just as important as standards. How the Internet won the "religious war" about standards in Scandinavia. *Knowledge, Technology and Policy*, Vol. 14 No. 3, Fall 2001, pp. 71- 89. Special Issue on IT Compatibility.

Hanseth, O., and Aanestad, M. (2003) Bootstrapping networks, infrastructures and communities. *Methods of Information in Medicine).*, 42: 384-391,

Hanseth, O., Ciborra, C., Braa. K. (2002) The Control Devolution ERP and the Side-effects of Globalization, *The DATA BASE for Advances in Information Systems.* Fall 2001/Winter 2002

Hanseth, O. and Lundberg, N. (2001) Designing Work Oriented Infrastructures. *Computer Supported Cooperative Work.*

Hanseth, O., and Monteiro, E. (1996). Information Infrastructure Development: The Tension between Standardization and Flexibility. In *Science, Technology & Human Values,* Vol. 21 No 4, Fall 1996, 407-426.

Hughes, T.P. (1983) *Networks of power. Electrification in Western society 1880 – 1930*. The John Hopkins Univ. Press.

Hughes, T.P. (1987) The evolution of large technical systems. In Bijker, W.E., Hughes T. P., and Pinch, T., *eds., The social construction of technological systems,*. Cambridge, MA: MIT Press.

Irmer, T. (1994) Shaping Future Telecommunications: The Challenge of Global Standardization. In *IEEE Communications Magazine. Special Issue on Standards: Their Global Impact.* Vol. 32 No. 1.

Kahn, R.E. (1994) The role of government in the evolution of the Internet. *Communications of the ACM*, 37(8):415–19. Special issue on Internet technology.

Katz, M., and Shapiro, C. (1985), Network Externalities, Competition and Compatibility. *American Economic Review*, vol. 75 (3), pp. 424-440.

Keen P. (1991):

Latour, B. (1991) Technology is society made durable. In J. Law, editor, *A sociology of monsters. Essays on power, technology and domination*, pages 103–131. Routledge.

Latour, B. (1999). *Pandora's Hope. Essays on the Reality of Science Studies.* Harvard University Press.

Leiner, B.M., Cerf, V.C., Clark, D.D., Kahn, R.E., Kleinrock, L., Lynch, D.C., Postel, J., Roberts, L.G., and Wolff, S.S (1997).The past and future history of the Internet. *Commun. ACM*, Vol. 40, No. 2, Pages 102-108.

Mathiassen, L., Munk-Madsen, A., Nielsen, P.-A., Stage, J. (2000). *Object-Oriented Analysis & Design*. Marko Publishing, Aalborg, Denmark

Monteiro, E. (1998). Scaling information infrastructure: the case of the next generation IP in Internet. *The Information Society*, 14(3).

Orlikowski, W. J. (1996) Improvising organizational transformation over time: a situated change perspective. *Information Systems Research*, 7(1):63-92.

Orlikowski, W. J., and Iacono, C. S. (2001). Research Commentary: Desperately Seeking the "IT" in IT Research -- A Call to Theorizing the IT Artifact. *Information Systems Research*, 12(2):121-134.

Parnas, D.L. (1972). A Technique for Software Module Specification with Examples. *Communications of the ACM*, Vol. 15 No. 5, pp. 330-336.

Porra, J. Colonial Systems. *Information Systems Research*, Vol. 10, No. 1, March 1999.

RFC (1994). The Internet standards process – revision 2. RFC 1602, IAB and IESG.

RFC (1995). The recommendation for the IP next generation protocol. RFC 1752, IAB and IESG.

RFC (2001a). Internet Official Protocol Standards, May 2001. IETF Standard #1 STANDARD. http:// www.cis.ohio-state.edu/cgi-bin/rfc/rfc2800.html

RFC (2001b) A SOCKS-based IPv6/IPv4 Gateway Mechanism. RFC 3089. http://www.cis.ohio-state.edu/ cgi-bin/rfc/rfc3089.html

Rolland, K. and Monteiro, E. (Forthcoming) Balancing the local and the global in infrastructural information systems. Forthcoming, *The Information Society*.

Shapiro, C., and Varian, H. R. (1999) *Information rules: a strategic guide to the network economy*. Boston, Mass.: Harvard Business School Press.

Sommerville, I (2001) *Software Engineering.* Addison-Wesley. 6th Edition.

Steinberg, S. G. (1995) Addressing the Future of the Net. *WIRED*, May: 141-44.

Weill, P., And Broadbent, M. (1998). *Leveraging the New Infrastructure*. Harvard Business School