Software development and IS use

Pekka Reijonen
reiska@cs.utu.fi
University of Turku/Laboris

Abstract

The differences in the world views of the two parties of organizational computing, computer scientists and information systems scientists, are rather large and seem to be persistent. There has been attempts to somehow merge or unify the field. In this paper we argue, however, that instead of a merger we need more definite borders around these domains. The two 'independent' domains (institutions) proposed are software development and IS use. We explore these institutions using Soft Systems Methodology and use the concept of institutionalization by Berger & Luckmann (1966) in interpreting the results. The conclusion of our analysis is that we should make a clear distinction between software development, which output is an artifact and an IS, which is cooperatively learned behavior, and choose our world view and methods as researchers and practitioners accordingly.

Keywords: software development, IS development, IS use, learning, SSM

1. Introduction

Even though Information Systems Science or equivalent is a discipline in most universities and business schools around the word, there are hundreds of books on Information Systems and tens of journals concentrating on Information Systems research there seems to be considerable problems in defining what exactly is the field of study of Information Systems Science. Even the formulation of a commonly accepted definition of the object of interest, the information system (IS) seems laborious (Orlikowski, 1992; Falkenberg et al.. 1998; Alter, 1999). After about ten years of work the IFIP WG 8.1 Task Group (FRamework of Information System COncepts, FRISCO), the aim of which was to develop "simple, clear and unambiguous definitions of, and a suitable terminology for the most fundamental concepts in the information systems field, including the notions of information and communication, and of organisation and information systems" (Falkenberg et al., 1998, p. 1) states in its final report: "The real concern (the misunderstanding of what is involved in organisational communication) is still there - in spite of our studies - and one may fear that some of the problems are innate to the various interested parties" (ibid., p. 2). The task group accounts these problems to the different "cultures" of computer science and social sciences and states that "The FRISCO group itself underestimated these problems, in particular the existence of 'hidden agendas' of the interested parties" (ibid., p. 2). The term culture refers here to the different scientific traditions and corresponds to scientific paradigm (Kuhn, 1970), community of observers or standard observers (Maturana, 1988), and subuniverse of meaning or conceptual machinery of universe maintenance (Berger &

Proceedings of IRIS 23. Laboratorium for Interaction Technology, University of Trollhättan Uddevalla, 2000. L. Svensson, U. Snis, C. Sørensen, H. Fägerlind, T. Lindroth, M. Magnusson, C. Östlund (eds.)

Luckmann, 1966).

These two cultures define information systems quite differently (Falkenberg et al., 1998, p. 5). Computer scientists interpret an information system as a technical system, implemented with computer and telecommunications technology and social scientists as a social system, such as an organisation in connection with its information needs. Both the technical and social system can, of course, be represented as conceptual systems on different levels of abstraction. Because of the different interpretations also the object of inquiry becomes different. Basing on Berger & Luckmann (1966) Nurminen (1988, p. 12) argues: "The rules, as it were, tell us not only how to act but also how to think - and what to think". With only some exaggeration we can argue that computer scientist concentrate on systems development, emphasize technology, and exclude nearby all actors from their analysis (or treat them as objects among other objects) whereas social scientists are inclined to study the implementation and use phases and emphasize the role of actors (users, human agency). This classification matches with the two views of Orlikowski (1992, p. 399) on the different scopes of technology found in the studies of technology, i.e. technology as 'hardware' and social technologies.

The differences between these cultures are not superficial, but are based on rather different philosophical positions and "Weltanschauung" (world view). In short, the interpretation of an information system as a technical system is based on the assumptions of Logical Empiricism (objectivity, dualistic ontology, and positivistic epistemology) whereas the proponents of the social system interpretation base their arguments on the Hermeneutic-Dialectic tradition (subjectivity, non-dualistic ontology, and hermeneutic epistemology) (see e.g. Radnitzky, 1970; Walsham, 1993). In information systems research, different authors have appointed different names to these approaches. For example, Nurminen (1988) uses three perspectives in describing the different approaches: systems-theoretical (bases on the tradition of logical empiricism and places the machine in the foreground), socio-technical perspective (basically a positivistic approach which attempts to take into account both the technological and human aspects), and humanistic (hermeneutic-dialectical approach which stresses the primacy of the human being). The different approaches have emerged chronologically after each other and according to Nurminen (1988, p. 17) "The chronological order is also reflected in content; each new perspective can be seen as a response to the challenges which earlier ones failed to satisfy". Orlikowski (1992, p. 399) has described the contents of the different phases as follows:

> "The early work assumed technology to be an objective, external force that would have (relatively) deterministic impacts on organizational properties such as structure. In contrast, a later group of researchers focused on the human action aspect of technology, seeing it more as a product of shared interpretation or interventions. The third, and more recent work on technology has reverted to a "soft" determinism where technology is posited as an external force having impacts, but where these impacts are moderated by human actors and organizational contexts".

The chronological order of the approaches leads easily to the interpretation that the "newer" approaches would somehow be better or even superior to the "older" ones. This kind of interpretation can be drawn at least from Nurminen (1988) and Orlikowski (1992). The order of the emergence of the different approaches is a historical fact, but at the same time it must be remembered that the new approaches have not replaced the older ones but just increased the number of the communities of observers. In other words, the older approaches are strongly alive and continually applied in the research of

technology. The coexistence of different approaches is found, however, disturbing. For example, the main goal of the FRISCO-group was to enhance the building of common terminology and definitions (Falkenberg et al., 1998). We argue, however, that the existence of different approaches is a positive thing and instead of trying to narrow the gap between them we should encourage the development of both main "cultures" and make the distinction between them more clear. Our main argument for the proposal is that the problems attacked by the two cultures belong to different institutional domains and hence, different approaches are needed.

The two domains examined here are the domain of software development and the domain of information systems use. They are sub-domains of the research and practice of the deployment of information technology in organizations, i.e., the metadomain of observation is about how the development, implementation and use of information technology can be conceptualized, studied, and carried out.

Table 1. The differences between the institutions of IS development and IS use on some distinctive attributes (variables). The attributes are produced by the author based on research literature (e.g. Maturana, 1988; Nurminen, 1988; Orlikowski, 1992; Falkenberg et al. 1998; Kling & Allen, 1996; Nissen, 1999).

Distinctive attribute (variable)	The institution of software development	The institution of IS use
Activity	Software development	Work tasks using the software
Goal of the activity	Software functioning according to specifications	Derived from the work (context dependent)
Evaluation criteria of the process	Time and monetary budget of the plan of the project	Efficient performance of work tasks
Evaluation criteria of the outcome	Technically functioning software (comparison of the technical requirements to the materialized software)	Work well done (comparison of outcome to local group norms)
Role of the IS	Outcome of the development process	Tool deployed in work, i.e. an environmental constraint
Type of IS entity	Composite entity	Simple entity (work contexts as a whole is a composite entity)
Basic philosophical stance of (most) actors and researchers	Logical empiricism	Hermeneutic-dialectical, non- or pre-theoretical
Standard observer	Software engineer	Professional worker in some vocational domain
Conception of world (reality)	Objective, mechanistic, rule-based, predictable, causal	Subjective, socially constructed, unpredictable, emergent
Conception of man	Rule-obeying, rational	Free will, creative
Process of institutionalization	Application of professional development methods	User learning

In her paper discussing the concept and role of technology in organizations using the structuration theory by Giddens (1984) as a framework Orlikowski (1992, p. 408) suggests that "… we recognize human interaction with technology as having two iterative modes: the *design mode* and the *use mode*". She makes, however, a reservation that the distinction is "… an analytical convenience only, and that in reality these modes of interaction are tightly coupled" (ibid., p. 408). We are in favor of a more clear distinction between the modes and support the position taken by Nurminen (1988, p. 15 - 16): information systems development and use (Nurminen (1988, p. 15) proposes a hierarchical system of institutions (institutional levels) consisting of *use of the system*,

systems development and implementation, and *information systems research*. In short, he justifies the hierarchical structure by stating that IS development methods are created by the institution of research and used in an institutionalized manner during development and implementation and so causing changes in the use institution. In our treatment in this paper the question of the hierarchy of the institutions is not crucial.) are treated as institutions in the sense of Berger & Luckmann (1966). Treating the phases as institutions means, among other things, that both institutions have different rules, actors, and activities as well as processes of institutionalization and standard observers (see Table 1). In short, they are different versa of the multiversa (Maturana, 1988).

The goal of a software development project is to bring about a computer based artifact which has the attributes and 'behaves' according to the rules specified in the requirements when installed on the specified technical infrastructure. In this process the future user is typically handled as an abstraction which has certain attributes like rationality and skill in and aptitude for behaving according to given rules - and the outcome of the process is usually called an "information system".

We argue, however, that there does not exist an information system before users are deploying the hardware and software as tools in their work tasks, i.e. it has reached the status of an institution. Hence, the concept 'information system' should not be applied to the bunch of hardware and software which is the result of a software development effort. Instead, the concept should refer only to the holistic entity which consists of (functioning!) hardware, software, knowledgeable users, real work tasks and processes, defined organizational structure and division of labor, etc. The development of an information system (IS) then becomes not the activity of software developers but the activity of users taking place after the installation of the software product. This development process is not a trivial task because information systems are artifacts, which have more or less profound effects on how their users can or must carry out their work tasks, can communicate with co-workers, are controlled, monitored, and rewarded, and what kind of vocabulary to use. Further, the implementation of an information system may change the division of labor, both between individuals and organizational functions inside the organization or even between organizations as in outsourcing. From the users' point of view all these effects are perceived as changes in the work environment. In order to adapt to the new environment users must be able to create a new, shared interpretation of their environment and, among other things, integrate the new tools into their work process. The adaptation process requires both unlearning, learning, creation of new knowledge, and acquirement of new skills.

The discussion presented in this paper is important and has implications both for research and practice. First, it aids in clarifying the field of IS research by making the distinction between computer science and information systems science explicit. Second, it gives reason and justifies the differences between the cultures in their basic ontological and epistemological stances, concepts, and methods. From these assumptions we can then deduct what kind of problems are, can, and should be studied on each of the fields - and how resources should be allocated in practical IS projects. Third, it elevates the role of users by maintaining that they are the ultimate developers of every IS in use. As a whole, the paper tries to convince the reader that the two institutions, software development and IS use, probably can not or at least should not, be combined. What is needed instead is that the differences are made explicit and all parties are aware of the presuppositions of others and their own.

We proceed by first giving a short presentation of the Soft Systems Methodology (SSM), which we then use in manifesting and discussing the differences

of the institution of software development and software use. In the discussion, we make use of the basic ideas of institutionalization as presented by Berger & Luckmann (1966) and compare our interpretation to that of Orlikowski (1992) who has based her study of the role of technology on structuration theory.

2. SSM and the institution of software development

We make use of the Soft Systems Methodology (SSM) developed by Checkland and his associates (Checkland, 1981, Checkland & Scholes, 1990, Checkland & Holwell, 1998) to further highlight the differences between the two institutions of computer based information systems, namely the software development and information systems use.

SSM is a general approach for making sense and giving structure to "real world" problems in order to make more informed proposals for their solution. Real world refers here to the perceived world in the same sense as Berger & Luckmann (Berger & Luckmann (1966) give the credit for this fundamental insight to Alfred Schütz, who throughout his work "... concentrated on the structure of the common-sense world of everyday life" (ibid., p. 27)) (1966, p. 27) define the main concern of the sociology of knowledge: "... what people 'know' as 'reality' in their everyday, non- or pre-theoretical lives". SSM is based on the ideas of General Systems Theory and hence the concept describing something which is 'a whole' is very essential. Even though the concept 'system' has been introduced as an abstract concept it is often also used to describe parts of the real world. In order to avoid confusion about what is reality and what is an abstraction of the reality the concept 'holon' is preferred to the concept 'system'. According to Checkland and Scholes (1990, p. 22) this concept is coined by Arthur Koestler in 1967 and means "constructed abstract wholes, conceding the word 'system' to everyday language and not trying to use it as a technical term" (ibid., pp. 25-26). The overall aim of SSM is to take seriously the subjectivity which is a fundamental characteristic of human affairs and to treat this subjectivity, if not exactly scientifically, at least in a way characterized by intellectual rigor (ibid., p. 30).

The basic model of the SSM methodology is presented in Figure 1. The process of the SSM inquiry takes place in two different realities; it begins and ends in the 'real world' whereas the models of the systems (holons) are created in the world of abstractions ('systems thinking about real world'). This borderline is intended to remind the actors about what the 'soft' in the name of the methodology means: the perceived world does not contain holons as 'hard systems thinkers' propose. According to Checkland & Scholes (1990, p. 22) the question if systems are 'abstract' or 'real' causes much confusion in the systems literature. In order to make their point clear they state "… it is perfectly legitimate for an investigator to say 'I will treat education provision *as if it were* system', but that is very different from declaring that it *is* a system" (ibid., p. 22).

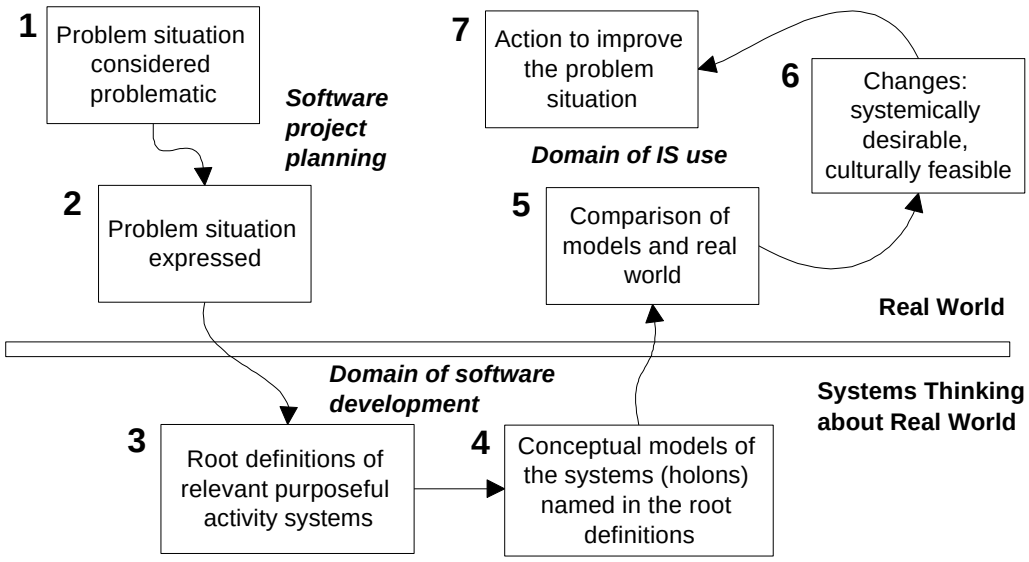

Figure 1. The conventional seven-stage model of Soft Systems Methodology (SSM), adapted from Checkland & Scholes (1990). The texts in *italics* added by the author.

According to our interpretation, when SSM is applied in IS development the phases 1 and 2 form the planning stage and requirements analysis, 3 and 4 the design and coding, and phases 5-7 implementation and use. If our basic interpretation is valid it has one very important consequence; the practice of software development is carried out in a conceptual domain outside the real world! We return to this notion in the discussion after examining the differences of the domains of software development and IS use applying the CATWOE analysis of SSM.

An essential tool in SSM for describing the problem situation is the CATWOE analysis. CATWOE is a mnemonic where C stands for Customers, A for Actors, T for Transformation process, W for Weltanschauung, O for Owners, and E for Environmental constraints. Our proposals for these entities in the domains of software development and IS use are presented in Table 2. It can be readily noted that the two domains differ from each other in all respects. For example while the software is the object of the transformation process in the domain of software development, it is an environmental constraint in the domain of IS use.

The crucial part of the CATWOE analysis is the formulation of root definitions. "A root definition expresses the core purpose of purposeful activity system. That core purpose is always expressed as a transformation process in which some entity, the 'input' is changed, or transformed, into some new form of the same entity, the 'output'"(Checkland & Scholes, 1990, p. 33). Our root definition for the institution of software development is the transformation process where *Software requirements* → *Software accepted by the customer* and for the institution of IS use *Installed software* → *IS in use*. We are aware that it is possible to create a great number of different root definitions - especially if the worldview is changed. As Maturana (1988, p. 7) has pointed out: "... there are as many domains of existence as kinds of distinctions the observer performs ... there are as many domains of truth as domains of existence she or he brings forth in her or his distinctions". We hope, however, that our root definitions do not belong to our private reality alone but can become a shared versum.

Table 2. The elements of SSM according to its CATWOE mnemonic in the domain of software development and IS use (Checkland & Scholes, 1990, p. 35).

CATWOE element	The institution of software development	The institution of IS use
Customers (the victims or beneficiaries of T)	Managers of software vendors' customers	Workforce and customers of software vendors' customers
Actors (those who would do T)	Developers (participating users are also developers)	Users, managers
Transformation process (the conversion of input to output)	Software requirements → Software accepted by the customer	Installed software → IS in use
Weltanschauung (the worldview which makes this T meaningful in context)	Professional systems developers create functioning software in time	A change of the IS in use will enhance production of products and services
Owners (those who could stop T)	Managers (of software vendors and their customer's companies)	Managers of software vendor's customer's companies
Environmental constraints (elements outside the system which it takes as given)	Technical artifacts, rules of logic, development tools, software requirements, resources (time, money)	Software (later IS), norms, rules, procedures

From root definitions (phase 3, Figure 1), conceptual models of the named holons are created (phase 4, Figure 1). This process is logic driven and if the root definitions are expressed properly in the XYZ form (*'a system to do X by Y in order to achieve Z'*, Checkland & Scholes, 1990, p. 36), the creation of the conceptual models is a rather straight forward process. We begin with the transfromation process of the software development where the *Software requirements* are transformed into *Software accepted by the customer*. When written according to the XYZ schema the root definition is *Convert the software requirements* (X) *by applying development methods* (Y) *to create a software product accepted by the customer* (Z). In other words, the activity taking place is software development carried out by software developers under the environmental constraints named in Table 2. As is evident from any book handling software engineering, several different conceptual models can be created from this root definition.

It is obvious that the human activity system described above is a 'holon' according to the definitions of SSM. In declaring their position between real the world and human activity systems (holons) Checkland & Scholes (1990, p. 22) state:

> "... it is perfectly legitimate for an investigator to say 'I will treat education provision *as if it were* a system', but that is very different from declaring that it *is* a system. This may seem a pedantic point, but it is an error which has dogged systems thinking and causes much confusion in the systems literature. Choosing to think about the world as if it were a system can be helpful. But this is a very different stance from arguing that the world *is* a system, a position which pretends to knowledge no human being can have".

In other words, human activity systems are holons, not 'real systems'. However, the outcome of the human activity system describing software development, the software product, has features which might justify us to say that it is a system - not just as if it were a system: a software running in a defined technical environment is a deterministic system the 'behavior' of which is totally predictable in the same way as a Turing machine: "… the operation of a Turing machine is completely determined by its functional matrix, so that two Turing machines with the same matrix are

indistinguishable as regards what they do" (Trakhtenbrot, 1963, p. 61). So, even though the *process* of software development is a holon the *object* of the process *is* a system, i.e. the conceptual model of the system is a complete, real world description of the system. This means that the rules of logic apply to this system and as a consequence,

- the behavior of the system can be exactly defined
- its behavior is always predictable
- the same input always produces the same output
- its output can be defined in measurable terms
- the congruity of its specifications (software requirements) with its behavior can be experimentally tested and rather exactly measured
- and so forth

Under these circumstances, it is a rather obvious choice to lean on the philosophical tradition of logical empiricism, which offers concepts and methods for performing the process of software development. In this 'world of conceptual models' there is no need for intensive interpretations of the results and the risk that different observers would make different conclusions is minimal - presuming that the basic assumptions are shared by the observers. In short, the positivistic methodology is an appropriate choice in most of the efforts (In our analysis, we have consciously neglected the first phases of the software development process, the decision to invest in IT and requirements analysis. Both of them are better understood as a social process where politics and power play a major role, i.e. they have many common characteristics with the institution of IS use.) performed during the process of software development. It must be remembered, however, that the output of a software development project is nothing more than a software product, an artifact with the pre-defined attributes.

3. The institution of IS use

Our root definition for the institution of IS use is *Installed software* → *IS in use*. In the XYZ schema it can be expressed, for example, in the following way: *Enhance user learning* (X) so that the *Installed software* (Y) becomes an institutionalized *IS in use* (Z). When this root definition is compared to the one given for the institution of software development it is readily seen that in this case competitive - and plausible - definitions are much easier to produce. Even though it is rather easy to produce a large number of different root definitions, detailed conceptual models seem to be nearly impossible to produce. According to our understanding, this has to do with the type of knowledge we have (and is possible to attain) from the field: when the knowledge about the 'laws' of human behavior (both individual and social) comes from an interpretation using a certain theoretical framework nearly every root definition seems plausible from the perspective of that framework. While discussing the different definitions of organizations found in IS literature Checkland & Holwell (1998, p. 70) note: "Such clear models are obviously helpful to inexperienced students, though they may make more experienced managers uneasy, since managers know how much of their time and energy is taken up, not with the substantive facts and the generic logic of their situations, but with the idiosyncrasies of interpretation of specific situations, and with the motivating myths and meanings which are as characteristic of organizations as the facts and the logic". Most of these root definitions can not, however, be transformed into a proper conceptual model, which could be compared to reality (see Figure 1).

However, as “each versum of the multiversa is equally valid” (Maturana, 1988, p. 7), we will proceed with our root definition and use the proposals of Orlikowski (1992) and the conceptual model of the institutionalization process proposed by Berger & Luckmann (1966) to discuss the transformation of an installed software to an IS in use.

As pointed out by Orlikowski (1992, p. 407), there is a time-space discontinuity between the design (The main finding of this paper, which is based on the structuration theory, is that technology has a dual nature; technology is physically constructed by actors working in a given social context, and technology is socially constructed by actors through the different meanings they attach to it and the various features they emphasize and use (Orlikowski, 1992, p. 406). This notion is actually the same as the main idea proposed by Berger & Luckmann (1966) except that they generalize the notion to the whole of our everyday reality: “... a world that originates in their thoughts and actions, and is maintained as real by these” (ibid., p. 33) and use of technology because development and use are accomplished in different organizations and the development of the artifact nearby inevitably precedes its use. As a consequence “users of technology often treat it as a closed system or a ‘black box’, while designers tend to adopt an open systems perspective on technology” (ibid., p. 407). Our interpretation is somewhat different: It is not question about a time-space discontinuity but about skills-knowledge discontinuity, i.e. users do not posses the necessary knowledge for opening the ‘black box’. This time-space discontinuity also contributes to the reification of technology thus hiding the human agency that initially produced the technology. Further, according to Orlikowski (ibid., p. 421) there exists the following causal link: “The greater the temporal and spatial distance between the construction of a technology and its application, the greater the likelihood that the technology will be interpreted and used with little flexibility”. [The term ‘interpretive flexibility’ is introduced by Orlikowski and refers to “the degree to which users of a technology are engaged in its constitution (physically and/or socially) during development or use” (Orlikowski, 1992, p. 409)].

The conclusions presented above may be valid from the theoretical perspective used by Orlikowski but the results of our SSM analysis in combination with the proposals of Berger & Luckmann (1966) lead to an other interpretation.

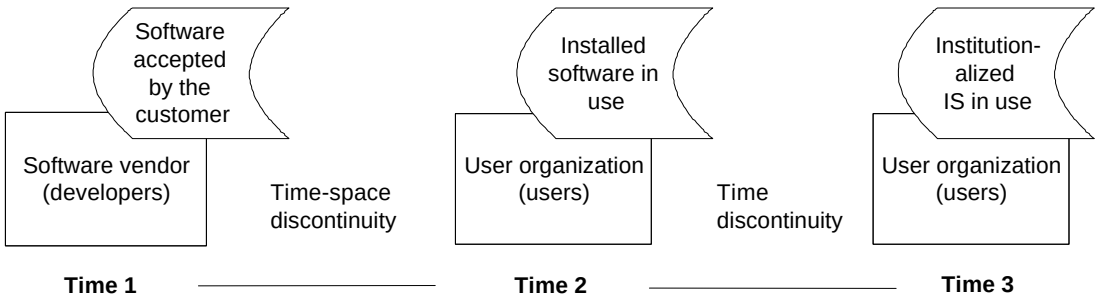

Figure 2. Time-space discontinuity of software development and IS use and the appropriate actors.

First, according to our SSM analysis the ‘outputs’ of the software vendor and the user organization are not the same: for the vendor the software is an object of work and for the user organization it should be a tool, i.e. it is a different human agency that has created the software and the institutionalized IS (c.f. the root definitions and Figure 2).

As a consequence, also their interpretations are different. Under these circumstances it is rather natural that parts of the software appear as a black box to the users, because they are interested in getting their work done, not to study computer science or programming. As Berger & Luckmann (1966, p. 57) have, introspectively, noted: "... a large part of the social stock of knowledge consists of recipes for the mastery of routine problems. Typically, I have little interest in going beyond this pragmatically necessary knowledge as long as the problems can indeed be mastered thereby".

Second, the separation of technological development from use may indeed enhance reification of technology, but we understand reification more as a result of a process than as the process itself. At least in this case, the process producing reification is user learning and it has to do with time but not with space. So, according to our interpretation, Orlikowski's claim that the time-space discrepancy between Time 1 and Time 2 (Figure 2) would have effect on the visibility of human agency, is false. Namely, the constructed nature of technology does not disappear because time elapses and the software is produced and used at different locations, but because users learn to use the software in their daily routines, i.e. the software is turned from a software product into a institutionalized IS. This learning process can not, however, begin before the Time 2 (Figure 2), i.e., before the software has been constructed. In this learning process, the space of construction is a rather irrelevant factor.

Third, the interpretative flexibility of technology does, indeed, diminish (If we are examining one, unchanged technology in one context. However, the interpretive flexibility of computers in general has greatly increased since the 1950s when computers were used for calculation - now computers are applied in every thinkable electronic equipment.) when time elapses (we do not see that the spatial distance would make any difference). The time count does not, however, begin from Time 1 but from Time 2, because the interpretive flexibility can not begin to diminish before the software is in use, can be learned, and become part of the objective reality of the users. So, the time discontinuity between Time 2 and Time 3 is of importance for the interpretative flexibility of the software. The reduction of the interpretive flexibility of a technology is not, however, a negative but a positive phenomenon: in practice, the coordinated use of an IS would not be possible if the interpretive flexibility of the IS would not diminish, i.e. the users must create a shared interpretation of the IS. As Maturana (1988, pp. 7 - 8) has noted, common knowledge is a presupposition of all coexistence of humans: "... disagreements between the observers, when they arise not from trivial logical mistakes within the same versum but from the observers standing in different versa, will have to be solved not by claiming a privileged access to an independent reality but through the generation of a common versum through coexistence in mutual acceptance. In the multiversa, coexistence demands consensus, that is, common knowledge".

The reduction of the interpretive flexibility takes place through user learning (Berger & Luckmann (1966, p. 70 - 72) prefer the concept 'habitualization' to learning. They may have chosen the concept habitualization because it refers as well to intentional as to unintentional learning. We prefer the concept 'learning' because its meaning is more clear.) and leads to the institutionalization of the IS. According to Berger & Luckmann (1966, pp. 70 - 77), when any action is repeated frequently it becomes cast into a pattern, i.e. becomes habitualized. When a group of people (e.g. users of a certain software) form a reciprocal typification of certain types of habitualized actions, the typification becomes institutionalized. These typifications are built up in the course of a shared history of the group of people. When institutionalized, the

typification begins to control human conduct irrespectively of the possible sanction mechanisms (the primary social control). It is important to note from this process description of institutionalization, that what gets institutionalized is not only the installed software, but also all other elements of work which are affected (changed) by the installed software. The number and type of affected variables is context dependent but typical elements are, for example, division of labor, work procedures, and control structures. Users who join the organization after the institutionalization of the IS have only a marginal effect on the earlier established IS institution and they are adapted into the institution through the process of secondary socialization. In secondary socialization actors internalize the 'reality' of some institution. Secondary socialization can be defined as "the acquisition of role-specific knowledge" (Berger & Luckmann, 1966, p.158). The (obvious) differences between the learning process of the users who create the institution and who become socialized into an existing institution is an interesting question but falls outside of the scope of this paper.

The institutionalization process of an IS begins, when a software product has been technically implemented and tested and people should begin to use it in their daily work tasks. According to the concepts applied in this paper, this means a shift from the institution of software development to the institution of information system use. Kling and Allen (1996) have called this phase organizational implementation. By introducing the term "organizational implementation" they want to highlight the position that implementation should not be considered as just coding a program but the effective use of computing requires further measures, which are social, psychological, and political rather than technical. Organizational implementation means "making a computer system accessible to those who could or should use it, and integrating its use into the routine work practices" (ibid., p. 269). This integration process is never an easy task and requires a shift of the perspective: "Organizational implementation is different from the strictly technical conception of implementation as coding a program" (ibid., p. 269). The way out of the problems encountered during the organizational implementation is according to Kling & Allen (ibid.) rather simple: the students of information and computer science shall be taught "… how organizations behave …" in order to avoid professionals "… who avoid working on both social and technological issues …" (ibid., p. 264). In their text, however, they do not make a clear distinction between when they are speaking about technical and social aspects of *artifacts in use* and when about the *development of artifacts*.

To us their solution (knowledge of technical principles and 'how organizations behave') does not, however, feel right. In the contrary, to us this is an explication of the principle of technological determinism (technological imperative): if the principles of technology and the principles of organizations were known we would be able to calculate and plan the organizational implementation in the same manner as the technical implementation. we argue, however, that if the situation would be this easy the solution would have been found, it would have become institutionalized, and replicated in all software development and implementation processes. This argument is supported by the fact that it is much easier to find descriptions of failures than successes in the research literature of IS (e.g. Landauer, 1995; Sauer, 1993; Star & Ruhleder, 1996). We think that even if failures are rather common in the 'IS reality' the overemphasis on failures has a methodological and epistemological grounding: It is easier to pick out some phenomena in a process, show that they are sub-optimal, and point out their possible causes than to show that a process is the best ever possible and show which *all* elements or factors are contributing to its excellence.

If our interpretation – the transformation process *Installed software* → *Institutionalized IS* is a learning process of the users – is correct, we should not spend all the resources on the technical infrastructure, software, or education of IS professionals. What is needed instead is that users have a reasonable chance and resources for turning the installed software into a institutionalized information system. In practice, this learning process can be promoted with user education and training, adequate support, and time.

4. Conclusions

The results from our SSM-based analysis of the institutions of software development and IS use suggest that it might be better to make the distinction between them clearer than to try to merge them. This conclusion is based on the notion that the scientific principles applied in natural sciences (logical empiricism) seem to support the efforts in the institution of software development whereas their applicability is rather limited in the institution of IS use, where the theories of social science (hermeneutic-dialectical) seem appropriate. This is in accordance with what Radnitzky (1970, p. 1) states in his analysis of the major contemporary schools of metascience: "Hermeneutic-dialectical philosophy has a metascience to the human sciences only. ... Logical empiricists have not developed a special metascience of the human sciences" (If we neglect the idea of unified science which means that the same principles are universally applicable to everything called 'scientific').

As noted by Nissen (1998, p. 194), the decisive point between the traditions is not between the methods of inquiry, but whether the object of study includes human beings or not. As a consequence, we should be flexible enough to change our basic philosophical assumptions according to the problem at hand, i.e. we should choose the philosophical stance which suits the solution of the problem rather than first choose a philosophical stance and then look what kinds of problems can be studied. Our proposal for a classification of the problem domain has two categories, software development with an artifact as an output and IS use, which is cooperatively learned behavior.

It seems that this distinction is gaining practical importance, because of the change from tailor made applications to off-the-self software, which is developed in 'software factories' according to the norms of industrial production. Future users have usually only a minimal role in this kind of production system and if users participate, they often come from a different culture (e.g. from the USA for software used in Europe). This distinction may also help to increase the relevancy of the research of the deployment of information technology in organizations, for example, in the field of information systems development research: According to the analysis of Iivari and Lyytinen (1998) there is a total of ten Scandinavian information systems development approaches and only two of them have gained some importance in practical use. These 'usable' approaches are 'the infological approach' and 'the formal approach' - and both of them have their theoretical roots in formal logic.

5. References

Alter, S. (1999). A general, yet useful theory of information systems. Communications of the Association for Information Systems, 1 (March 1999), Article 13. An electronic journal

available at http://cais.aisnet.org/contents.asp, (30.3.2000).
Berger, P. L. & Luckmann, T. (1966). The social construction of reality. A treatise in the sociology of knowledge. London: Penguin Books.
Checkland, P. B. (1981). Systems thinking, systems practice. Chichester, England: John Wiley & Sons.
Checkland, P. B., & Scholes, J. (1990). Soft Systems Methodology in action. Chichester, England: John Wiley & Sons.
Checkland, P. & Holwell, S. (1998). Information, systems and information systems - making sense of the field. Chichester, England: John Wiley & Sons.
Falkenberg, E. D., Hesse, W., Lindgreen, P., Nilsson, B. E., Oei, J. L. H., Rolland, C., Stamper, R. K., Van Asshe, F. J. M., Verrijn-Stuart, A., & Voss, K. (1998). A framework of information system concepts (The FRISCO-report, Web-edition), IFIP, available by anonymous ftp://ftp.leidenuniv.nl/pub/rul/fri-full.zip, 20.4.1999.
Giddens, A. (1984). The constitution of society: Outline of the theory of structure. Berkeley, CA: University of California Press.
Iivari, J., & Lyytinen, K. (1998). Research on information systems development in Scandinavia - Unity in plurality. Scandinavian Journal of Information systems, 10 (1&2), 135 - 186.
Kling, R. & Allen, J. P. (1996). Can computer science solve organizational problems? The case for organizational informatics. In R. Kling (Ed.), Computerization and Controversy (2nd edition) (pp. 261 - 276). New York: Academic Press.
Kuhn, T. S. (1970). The structure of scientific revolutions. Chicago: .University of Chicago Press.
Landauer, T. K.. (1995). The trouble with computers - Usefulness, usability, and productivity. Cambridge, Mass.: The MIT Press.
Maturana, H. R. (1988). Ontology of observing: The biological foundations of self consciousness and the physical domain of existence. In R. Donaldson (Ed.), Texts in cybernetic theory: An in-depth exploration of the thoughts of Humberto Maturana, William T. Powers, and Ernst von Glaserfeld. A conference workbook: American Society for Cybernetics.
Nissen, H-E. (1998). Quo vadis: Scandinavian information systems development research? Scandinavian Journal of Information Systems, 10 (1&2), 193 - 204.
Nurminen, M. I. (1988). People or computers: Three ways of looking at information systems. Lund: Studentlitteratur.
Orlikowski, W. J. (1992). The duality of technology: Rethinking the concept of technology in organizations. Organization Science, 3 (3), 398 - 427.
Radnitzky, G. (1970). Contemporary schools of metascience. Vol. II: Continental schools of metascience. Göteborg: Akademiförlaget.
Sauer, C. (1993). Why information systems fail: A case study approach. Henley-on-Thames.: Alfred Waller Limited.
Star, S. L. & Ruhleder, K. (1996). Steps toward an ecology of infrastructure: Design and access for large information spaces. Information Systems Research, 7 (1), 111 - 134.
Trakhtenbrot, B. A. (1963). Algorithms and automatic computing machines. Boston: D.C. Heath and Company.
Walsham, G. (1993). Interpreting information systems in organizations. New York: John Wiley.