Alfred Holl
Felix Paetzold
Robert Breun

Cooperative, cyclic-iterative knowledge gain in information systems anti-aging

Georg Simon Ohm University of Applied Sciences
Nuremberg, Germany
2011

ISBN 978-3-00-034697-2

ABSTRACT

This study shows a new way of aligning the evolution of an organization's information system with its own evolution: a methodological guideline for IS consultants is built upon an explicit epistemological basis. In the first part, given the epistemological properties of an organization (a psycho-social, time-sensitive environment), the need of a close cooperation between organization experts and IS experts ("observers" with different points of view) is underlined; "soft" skill training strengthens the effectiveness of communication between the two different perspectives in evaluating the "world" of an organization. A permanent cyclic-iterative knowledge gain is envisaged via the cooperation and convergence of different "scientific disciplines" and "approaches" that enable multi-level learning within the organization. Such learning constitutes a core method of information systems design, development and maintenance. In the second part, information systems are classified and the term *information systems aging* is defined. According to Lehman's laws, a lack of appropriate anti-aging methods leads to an enormous increase in information systems complexity and, in consequence, to a complete loss of usability and maintainability of the information systems affected. Therefore, specific methods against information systems aging are presented for the different phases (analytic, synthetic and maintenance phase) of the information systems life cycle. All of these methods refer to the aspects of temporal dynamics of organizations and / or common responsibility of organization experts and IS experts and can be visualized by cycles of knowledge gain (*mayeutic cycles*).

Keywords: change management, cognition, creeping requirements, epistemology, E-type system, information systems aging, mayeutic cycle, multi-perspectivity, requirements engineering, software maintenance

PREFACE OF THE SENIOR AUTHOR

This research report covers the wide range from the epistemological foundations of information systems anti-aging to selected anti-aging methods for the practical use in organizations. Therefore, it addresses researchers and senior students in the field of information systems as well as practitioners in business and administration.

The study is the extended version of two master's theses in information systems written by Felix Paetzold and Robert Breun at the Georg Simon Ohm University of Applied Sciences of Nuremberg, Germany, in spring 2007.

I would like to express my acknowledgments to

- Werner Kügel, University of Applied Sciences of Nuremberg, for checking the English manuscript
- Hans-Erik Nissen, University of Lund, Sweden, for fruitful discussions during the initial phase of this research project

Nuremberg, Germany, July 2011

Alfred Holl

Alfred.Holl@ohm-hochschule.de

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview of structure, aims and reasoning

A lot of people would probably think of plastic surgery when they were asked about the term “anti-aging”, and hardly anyone would think of software or information systems. There are two main reasons why the authors decided to write about information systems anti-aging although it is not very popular in IS theory (in this study, the scientific discipline *information systems* is always abbreviated to *IS*).

The first one is that we are convinced of the importance of a discussion about information systems anti-aging. If no anti-aging methods are applied during development and maintenance, the life cycles of information systems will be shortened extremely and, as a result, high costs and great efforts will be required for the design and development of completely new ones.

The second reason is that the mayeutic cycle as the general epistemological core model of knowledge gain can be applied in an excellent way to describe background and solutions of information systems anti-aging.

With regard to organizations and their information systems, knowledge gain has a cooperative and a cyclic-iterative aspect as organizations (with the term *organization* comprising any profit and non-profit organizations in industry and administration) are psycho-social and time-sensitive environments. Our two main lines of argumentation are based upon these two essential properties of organizations. Organization experts and IS experts have to cope with the multi-perspectivity of their different views (cooperative aspect in Sections 2.1.1 and 2.2.1) as well as with the temporal dynamics of organizations during information systems maintenance (cyclic-iterative aspect in Sections 2.1.2 and 2.2.2).

To learn how to deal with these two issues will contribute a lot to solve the question of what can be done against information systems aging (Section 3). All of the methods presented will include the cooperative and the cyclic-iterative aspects.

Although the philosophic terminology of approaches like ours is not common, it would be advantageous to make IS experts aware of this background and to encourage further research in this field. Therefore, we emphasize a strong and deep theoretical foundation in the following section.

1.2 Epistemological and terminological remarks

Within the field of epistemology, also called theory of knowledge, the fundamental questions about human knowledge acquisition (origin, nature, limits) are discussed. Karl Popper systematized the discussion with his ontological model of three levels of existence (ontological levels) for objects of cognition (Popper, 1972): a real world (*World 1*, reality of organizations), a world of subjective experience (*World 2*, the mental models of an organization expert or an IS expert) and a cultural world (*World 3*, documented models and human artifacts, such as information systems) (Figure 1).

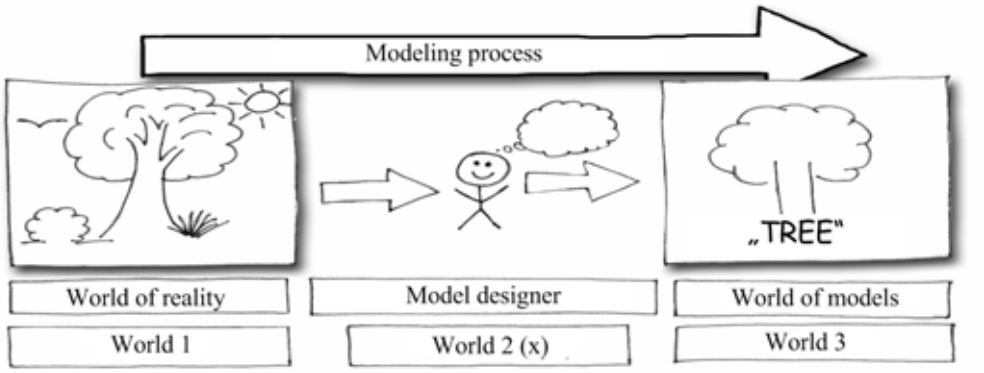


Figure 1: Modeling process from the view of realism (adapted from Holl & Krach, 2002, p. 54)

We do not refer to Popper's critical rationalism, but only to his theory of the three worlds as an epistemological meta-model. His assumption of World(s) 2 is closely related to basic considerations of *hermeneutics* (the interpretation of the meaning of texts and works of art) and *phenomenology* (studies everyday life experiences and perceptions of individuals) because *phenomena* as personal experiences are assigned to World 2. With regard to phenomenological methods, we confine ourselves to basic principles, such as Heidegger's statement that "phenomenological description as a method lies in interpretation" (Heidegger, 1962).

On the whole, our approach is eclectic. Its epistemological basis is composed of elements from critical realism, moderate constructivism and evolutionary epistemology, the latter connected to the names of Konrad Lorenz, Rupert Riedl and Gerhard Vollmer. All of these researchers consider older philosophical traditions in the light of modern biology. A couple of years ago, one of us (AH) already successfully transferred some of Riedl's thoughts to IS

(Holl, Krach & Mnich, 2000, pp. 204-208) and we would like to do that again in this study.

As epistemological approaches cannot generally be discussed within the scope of this study, we will follow the brief argumentation in Holl & Krach (2002, pp. 54-55), adopt Karl Popper's three worlds and assume the existence of a world of reality outside of the human consciousness (cf. Schmidt, 1982, p. 572) that exists independently of human perception. Epistemologically, this standpoint is called realism. Its two main variants, the naïve one and the critical one, will now be characterized as well as moderate constructivism and, as an explanatory approach for critical realism, evolutionary epistemology (for more details cf. Holl & Maydt, 2007, pp. 42-49).

Naïve realism: there exists a real world; it is such as we perceive it (Vollmer, 1990, p. 35). This view is well suitable for small problems of everyday life, but encounters its limits as soon as we leave the everyday area and advance into areas with 'very small' objects (e.g. quantum dynamics) or areas with 'very large' objects (e.g. astronomy) or deal with complex problems, such as the modeling of socio-technical structures as in IS. Keeping the assumption that a real world exists, the legitimate critique of naïve realism (e.g. due to optical illusions, Heisenberg's relation of uncertainty) leads to critical realism.

Critical realism (cf. Bhaskar, 1989): there exists a real world; but it is not completely such as we experience it (Vollmer, 1990, p. 35). The critical realist assumes that subjective distortions influence the human perception of real world objects. Critical realism is not far from *moderate constructivism* (cf. Holl & Maydt, 2007, p. 45; Goorhuis, 1994) which, as critical realism, does not consider humans as passively perceiving beings, but as beings who construct their world(s) on their own. On comparison with critical realism, the constructed part of human knowledge is emphasized more strongly.

Evolutionary epistemology: we can assume that the subjective structures of pre-scientific experiential cognition, including the (neural) structures of perception, are adapted to the environment where they were developed. Nevertheless, it cannot be expected that these cognitive structures are suitable for all real structures, nor that they are adequate to perceive all these structures correctly (Vollmer, 1990, p. 162). Due to evolution, the cognitive structures of humans are limited and only partially universal. Therefore, reflection about the functionality of the human cognitive equipment is necessary if one goes beyond everyday world, that is, if we leave the area in and for which the human cognitive strategies were selected in the course of evolution (Vollmer, 1990, p. 161).

Before this background, we will now discuss the use of the term "system" according to the considerations in Holl (1999, pp. 192-194).

Decomposition (structuring) is a prerequisite for a reduction of complexity. The latter is necessary for any knowledge. Humans have to structure their World 1 images in order to reduce the latter's complexity. They have to divide the images into a lot of components; otherwise their human cognitive power would not be able to cope with them. It is quite natural that the procedure of decomposition destroys interdependencies between the components.

Therefore, humans try to use their "cognitive scissors" only in those places of their World 1 images where they suppose only a few connections which they consider as negligible from an idealistic point of view. In a naïve-realistic manner, they retransfer this assumption to World 1. More or less arbitrarily, they delimit components / segments (which, for instance, can encompass processes, items, information), isolate them from their interdependencies and "cut" them out artificially. Thus, a dilemma arises which we call the *problem of isolation* (cf. Holl, 1999, pp. 192-194). Humans have to acquire knowledge about World 1, but they can do this only via decompositions which, on the other hand, neglect interdependencies.

Humans construct *systems* (mental representations of World 1 segments) which become part of World 2 or World 3 when they are described verbally. Humans believe in a naïve-realistic way that systems belong to World 1. But from our point of view, systems are special kinds of models, descriptive categories which are to approximate immanent categories of World 1. Our interpretation of systems is similar to Peter Checkland's (1981, pp. 162-166) who has chosen to reserve the term "system" for what he calls "notional systems", i.e. what somebody (or some group of persons) in some context for some reason chooses to look upon as a system.

Structuring is continued on the following lower level. Humans give an internal structure (order) to systems by dividing them into *interacting components* (therefore, a system is more than the sum of its components). The *internal connections* (between the components) are assumed to be stronger than the *external connections* (of the entire system to its surroundings). The components can now be interpreted as (sub- / partial) systems in their turn. The decomposition process continues on diverse abstraction levels.

It is true that there are *no natural, closed (self-contained) systems*, that are completely separated from their environment and do not have any kind of interactions and exchange with it. There must, however, be segments of World 1 ("*system-like structures*") with strong internal connections and weak external ones. Otherwise the cognitive use of segmentations of that kind would have been eliminated during evolution (this is an argument from evolutionary epistemology). The starting point of the system concept can be found in optical-tangible items, whose visual contours coincide with their tangible boundaries. Physically spoken, they are *solids* (for instance apples, stones) with

very strong internal connections and comparably weak external ones. They can be moved in relation to other items. Items of that kind can be comprehended in a naïve-realistic way. Starting from this type of objects of cognition, the system concept is transferred to other objects of cognition, for instance organizations or departments, where it is only applicable with the necessary restrictions and modifications.

Based upon the understanding of systems mentioned, we can say that an *open system* is a descriptive entity (in World 2 or World 3) representing a "system-like structure" of World 1 where exchanges (of information, temperature, chemical substances etc.) with the surroundings are considered as important by some person in some context. A *social system* is always an open system which describes a group of living organisms, especially of humans.

Ending this sequence of terminological explanations, we use the term *organizational information system* for an open information-processing system with or without IT support. A *technical information system*, briefly *information system*, is the software part (technical system) of an organizational information system.

To label people working with information systems, we drop the traditional word *user* and replace it with the more adequate term *organization expert*; this is already done in the essential book by Steinmüller (1993, p. 168). From our view, an organization expert is every human in an organization who is somehow affected by an information system – a worker as well as the chief executive officer. The authors are aware of the fact that there are organization experts in different functions and on different hierarchical levels (cf. Hirschheim & Klein, 2003, pp. 266-267) whose responsibility, knowledge and participation regarding development and use of information systems are very different. As the differences between the various kinds of organization experts, however, are not in the focus of this study, using the umbrella term *organization expert* allows better argumentation. Therefore the main distinction in this study remains the one between organization experts and IS experts, that is, between persons inside and outside the organization who have to cooperate in developing and maintaining the organization's information system.

2. INFORMATION SYSTEMS EVOLUTION AS PROCESS OF COOPERATIVE, CYCLIC-ITERATIVE KNOWLEDGE GAIN

2.1 Epistemological properties of organizations as IS *observanda*

Organizations and parts of them (departments) as the objects of cognition (*observanda*) in IS are segments of reality that can be described as dynamic, open, psycho-social and information-processing systems:

- *psycho-social* as they comprise humans who work together in a social context as well as
  open and *information-processing* as they exchange (send and receive) information with their environment; this property (Section 2.1.1) will lead to the cooperative aspect of permanent knowledge gain (Section 2.2.1)
- *dynamic* as their behavior and structure may change over time; this property (Section 2.1.2) will lead to the cyclic-iterative aspect of permanent knowledge gain (Section 2.2.2)

This is not a complete list (cf. Holl, 1999, pp. 194-198; Holl & Maydt, 2007, pp. 40-41) of an organization's epistemological properties, but only mentions those relevant in the context of information systems anti-aging.

In contrast to *observanda* in natural sciences, objects of cognition in social sciences, such as organizations, cannot only be observed. The human beings constituting an organization have the ability of verbal communication and thus can inform – in a subjective way, of course, not in an objective way – the observer about their activities and their mental models (cf. Section 2.1.1; cf. Holl & Maydt, 2007, pp. 33-34).

This is just the point where hermeneutics becomes important in IS (cf. Section 2.2.2) as the statements of organization experts and IS experts have to be interpreted and understood mutually. Therefore, on the one hand, a mere functionalist and technological view is completely misleading. On the other hand, however, we consider it a false axiom to assume that the observation of an organization could be confined to hermeneutic methods as the organization experts would know everything best about their organization. This can be correct for some well-managed and well-structured organizations, but it is not correct for ordinary and especially for chaotic organizations which cannot successfully cope with the needs of the market without any cooperation of consultants and IS experts. Therefore, observation in IS has to include – at least

– the observation of human informational products, such as customer lists, orders, invoices, statistic lists, statements of account etc., the optimization of business processes and the transfer of reference knowledge from other organizations in order to finally design adequate formal data and process models (cf. Section 2.2.1).

This is the reason why hermeneutics is not sufficient to describe the knowledge gain regarding organizations. Due to the two special features of organizations (mentioned above), observation in IS has a lot in common with observation in social sciences although it goes beyond a mere observation of human behavior in a social context and a mere hermeneutic interpretation of statements of organization experts.

Therefore and according to our personal background, we prefer observation in natural sciences as a starting point for a characterization of observation in IS. Furthermore, ethology (scientific study of animal behavior; can be extended to human societies) and psychology (both considered as natural sciences) are closely related to many aspects of sociology as well.

Compared to objects of cognition in natural sciences, there are, according to the two properties mentioned above, at least two important aspects of organizations which have to be considered in the area of information systems anti-aging:

- the mutual influence of observer and *observandum* (cf. Section 2.1.1)
- the temporal dynamics of organizations (cf. Section 2.1.2)

2.1.1 Mutual influence of observer and *observandum* in organizations

As organizations are psycho-social entities, their behavior can be influenced by observation and observers, similar to the behavior of sub-atomic particles in modern physics (the properties of which are completely different from those of Newtonian physics) or the behavior of animals in ethology. IS experts as observers cannot avoid this influence, but they have to be aware of it because they cannot and shall not hide themselves.

For instance, employees can be afraid of being fired, of a new IT system or of changes in general; others look forward to being supported by a new IT system as soon as possible or have false expectations towards it etc. All of these emotions exert an influence on the behavior of employees in the organization observed.

If an organization expert, interviewed by an IS expert, notices that he could perform some of his tasks more efficiently, one can imagine the following reactions (cf. Holl & Maydt, 2007, p. 42; Holl, 1999, p. 205):

- The organization expert changes his activities, but does not inform the IS expert about the modification. The latter still knows the old ones.
- The organization expert informs the IS expert about possible improvements, but keeps his old business processes unchanged.
- The organization expert intentionally describes his activities in a palliative way, e.g. in order to hide inefficiency.
- The organization expert totally refuses to collaborate with the IS expert.
- The organization expert tries to act better than usual for a short time, but then returns to his old behavioral patterns.

The cartoon in Figure 2 is a good example for avoiding the observer's influence on the object of cognition. It shows a mole that is observed by an ethologist. By using a self-made molehill as camouflage, the ethologist hopes that he can observe the natural behavior of the mole without disturbing it. Whether the mole and its behavior are still affected by the presence of the ethologist or his camouflage (which in this case would counteract its intended purpose) or not, remains uncertain (cf. Holl & Maydt, 2007, pp. 37-38); e.g. the ethologist can destroy tunnels of the mole by digging his observation hole etc.

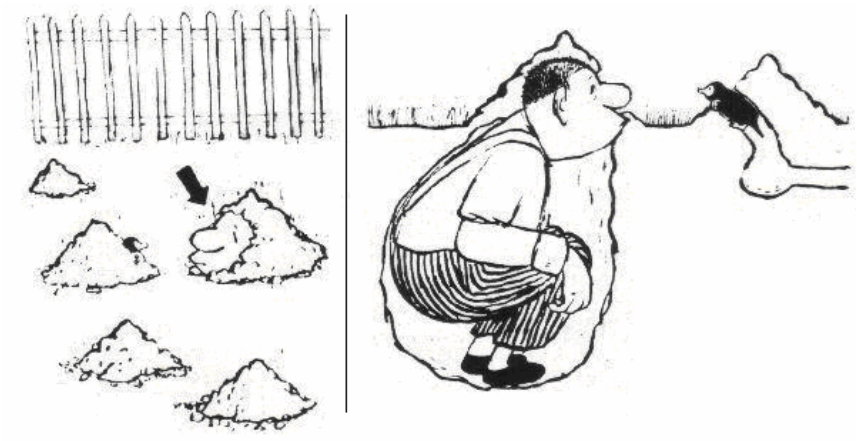


Figure 2: Ethologist and mole

(cf. Holl & Maydt, 2007, p. 38;
adapted to IS by Schmidt, 1993, p. 12 from Bülow, 1968, p. 19)

However, the cartoon can also be interpreted the other way round. By observing the mole for a long time, the ethologist seems to adapt to parts of the mole's behavior. For instance, the posture of the ethologist in his hole is very similar to the mole's posture. Particularly the posture of the ethologist's hand matches the posture of the mole's claw.

This aspect shows that not only *observanda* can be influenced by observers, but, vice versa, observers (IS experts) can be influenced by their objects of cognition, that is, by organization experts.

At least two different kinds of influence by organization experts on IS experts can be distinguished:

- influence via language communication
- influence via transport of emotions

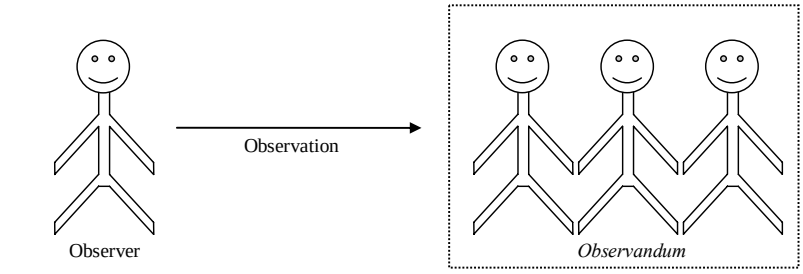
The organization experts can inform about themselves and their activities (cf. Section 2.1). Interpretations of an organization depend on their presentation. As a consequence, an eloquent employee can certainly explain his opinion of the organization more clearly and more effectively than a taciturn and shy employee and thus can more strongly influence an IS expert.

Depending on the personal relationship between the IS expert and the organization expert, the information flow is affected positively or negatively due to emotions: In case of antipathy, a tense atmosphere will predominate in a conversation. The IS expert is less motivated and does not adequately take the organization expert's statements into consideration. If mutual sympathy predominates, however, the conversation will take place in a relaxed atmosphere. The IS expert is better motivated, shows more interest in the organization expert's arguments and, therefore, can solve open problems more easily (cf. Holl & Maydt, 2007, p. 42).

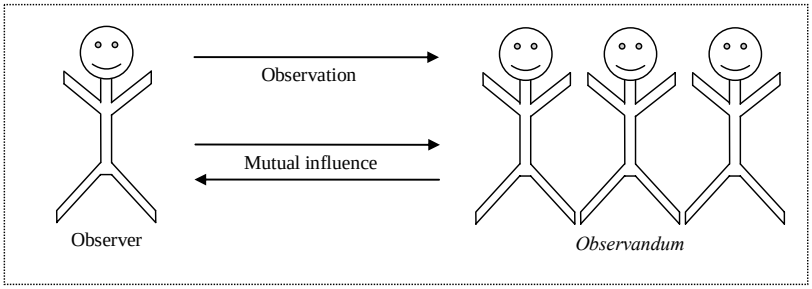
So after all, it is evident that the subject-object (observer-*observanda*) separation cannot be conserved, neither by an ethologist nor by an IS expert.

As almost any kind of observation within an organization is detected by someone and, furthermore, hidden observation often is at odds with laws protecting the employees' rights, a separation of that kind is almost impossible in the reality of an organization. The organization will always realize observers and observations with the consequence of mutual influence. Even if the subject-object separation were intended (IS experts should not do that!), it could not be achieved.

These considerations lead to the insight that subject and object of cognition in IS are not strictly separated from each other as a naïve-realistic point of view (Figure 3, 1.) would assume, but together form a new structure in which the observation is done (Figure 3, 2.). Therefore, we can say that the observer (like the ethologist) does not simply observe the organization's behavior, but, more exactly, that he – influenced by the organization – observes how the organization behaves under his observation (cf. Holl & Maydt, 2007, pp. 37-38). From a critical-realistic view, an observer and his *observandum* always build a new structure that can be described as a new system consisting of observer and *observandum*.



1. Naïve-realistic view: separation of observer and *observandum*



2. Critical-realistic view: mutual influence of observer and *observandum* described as a new system consisting of observer and *observandum*

Figure 3: Mutual influence of observer and *observandum*

This argumentation will be continued in Section 2.2.1.

2.1.2 Temporal dynamics of organizations

Organizations are temporally dynamic. Due to internal and external reasons, they can change their behavior and thus affect information systems and cause changes in them and, therefore, inevitably the aging of information systems.

If an organization is forced by law to collect additional data about its business activities, we encounter an external reason for the demand to integrate a new module in the organization's information system. An example for an internal reason is a new management that changes the organization's structure. In this case, the information system has to be adapted to the new structure.

This dynamics of objects of cognition does not occur in Newtonian physics (some stone will "always" behave in the same way), but in sub-atomic particle physics (wave-particle dualism) as well as in ethology and psychology where the behavior of individuals and social groups is dealt with.

The temporal dynamics of organizations requires an adequate information system life cycle model, such as the exemplary one in Table 1.

The life cycle model in Table 1 can be applied to the development of individual software and to the customizing of standard software. In an initial step, a first version of an information system is developed which is later on maintained in many iterative steps.

Design step	Main phase	Sub-phase	Model level	Model purpose
First design and maintaining redesigns	**Analytic phase: problem analysis**	**Elicitation of the current state of the organization**	**Information-relevant models**	**Descriptive models**
		Analysis of the current state of the organization		
		(Re-) Design of the planned state of the organization (LOCK)		**Prescriptive models**
		(Re-) Design of the business concept of the information system (KEY)		
	Synthetic phase: IT system development and use	**Design of the technical concept of the information system**	**Implementation-relevant models**	
		Programming		
		Test		
		Use in an organization	**Information-relevant models**	

Table 1: Information system life cycle model

(adapted from Holl, 1999, p. 199)

The aim of the initial step is the first design of an information system. To achieve this aim, an analytic main phase must be passed. There, the current state of the organization is elicited and analyzed using descriptive information-relevant (without respect to IT details) models which aim at the detailed understanding of a problem. Still in the analytic phase, the planned state of the organization (*lock*) and the information system's business concept (*key*) are designed using prescriptive information-relevant models which aim at the construction of conceptual models (cf. Holl, 1999, p. 199).

The principle of *key* and *lock* visualizes the situation of an information system and its application area in an organization. They have to perfectly fit together like key and lock. To achieve this aim, good co-operation between IS experts and organization experts is inevitable. From this follows their common responsibility for an adequate design of the planned state of the organization as a prerequisite for an efficient design of the business concept of the information system (cf. Section 2.2.1).

After the analytic phase has been completed, the synthetic phase has to be started. A first version of the technical concept of the information system is designed, programmed and tested using prescriptive implementation-relevant (with respect to IT details) models which aim at the construction of an IT system. Then, the information system is installed and used in an organization.

The initial step of the first design is followed by iterative steps of maintaining redesign which in turn consist of exactly the same phases and subphases as the initial step.

This argumentation will be continued in Section 2.2.2.

2.2 Knowledge gain in organizations

2.2.1 Cooperative knowledge gain: multi-perspectivity of IS experts and organization experts

Regarding the background outlined in Section 2.1.1, we now explain the cooperative aspect of permanent knowledge gain in organizations. Due to its temporal dynamics, an organization can change its behavior and – in succession – make a redesign of the business concept of its information system (*key*) neccessary. According to the principle of key and lock, this automatically leads to the consequence that the design of the planned state of the organization (*lock*) also has to be adapted to the new conditions.

Therefore, the employees in the organization observed carry a great responsibility to inform the IS experts about their organization (and its business processes, information flows, data etc.). This is why the employees have to be treated (by the IS experts) as organization experts and not only as IT users.

Figure 4 shows the different perspectives of an IS expert and an organization expert on identical segments of reality. An organization expert is not able to communicate with an IS expert from the very beginning because of their different points of view in relation to one and the same issue.

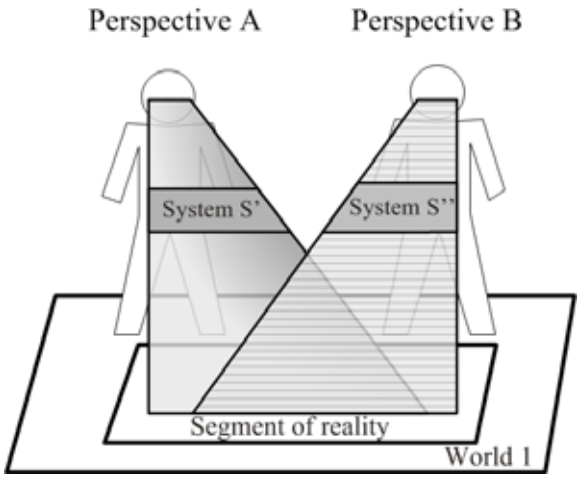


Figure 4: Systems are relative to perspectives

(adapted from Steinmüller, 1993, p. 168)

Therefore, in order to avoid misunderstandings which will cause serious problems in the design of information systems (cf. Figure 5), special education and training is required for organization experts. In this context, we do not aim at a simple and trivial "data / IT maturity" of an organization and its experts, but at general abilities of employees – soft skills not depending on IT.

Among others, the following types of abilities have to be trained and developed:

- abilities related to language communication
  - ability to avoid organization-internal terminology (idiolects)
  - ability to express and present one's opinion clearly (rhetoric)
- abilities related to working in teams
  - ability to carry one's point in discussions

    - ability to work successfully together with experts of other disciplines
    - ability to avoid and / or solve social conflicts within teams
- abilities related to reflection and logical thinking
    - ability to avoid implicit assumptions
    - ability to separate standard cases from special / marginal cases
    - ability to develop some sensitivity for logical problems
- abilities related to abstraction
    - ability to abstract from special cases
    - ability to abstract from one's own point of view
    - ability to abstract from one's own interests

Our opinion is not far from participative strategies although it clearly underlines the organization experts' responsibility and thus goes far beyond mere participation.

Vice versa, IS experts need special epistemological and communicative trainings as well. The necessity of bilateral communicative training is taken into consideration in modern requirements engineering approaches, such as Rupp (2004) (cf. Section 3.2.1).

Since the IS experts carry a great responsibility as well, we can talk of a joint responsibility. Each of the two responsibilities, however, is different as the different types of experts have different knowledge. Organization experts have knowledge about structures in their organization, but do not know everything about it. Therefore, the knowledge of organization experts cannot be the only source for the design of a future information system. Their knowledge has to be completed with IS experts' knowledge about formal-mathematical structures, improvement of business processes, information processing, software and hardware technologies (cf. Section 2.1). These knowledge aspects have to be integrated for the benefit of the organization so that there is joint knowledge gain (about the organization) as well.

During this bilateral knowledge acquisition, IS experts should learn a lot about the organization. One of us (AH) is used to provoking IS experts when he requires that they should become so familiar with the application area of their information system that they could professionally work with it in an organization.

Figure 5 shows the consequences of misunderstandings between different experts within a project. If the experts are not able to or do not want to build bridges between their different points of view, their knowledge aspects cannot be integrated to benefit the project, but conflict with each other which, in consequence, leads to a flop of the entire project.

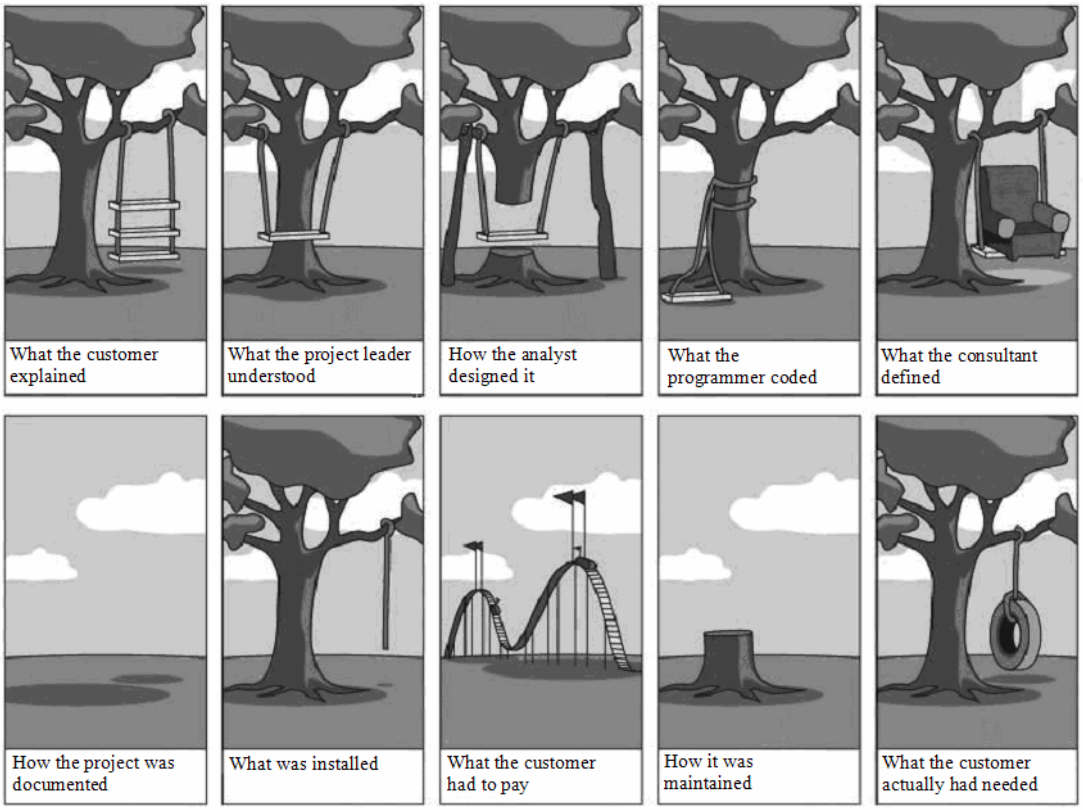


Figure 5: Multi-perspectivity in project management (*passim* on the Internet)

Members of project teams have to know that there is not only one "correct" point of view, but many different points of view that allow interpreting the same issue from various different perspectives. The more different points of view are taken into consideration, the better the understanding of an issue will be.

In Figure 6, we see a lot of persons, each one with a special personal background and a special perspective of the forest they are in. Each statement is totally correct with regard to the particular person's point of view although the statements do not have anything in common. The best description of the forest can indeed be derived by considering the entirety of all of the statements.

Figure 6: "Who sees the forest?"

(Holl & Maydt, 2007, p. 37; adapted from Hajos, 1991, p. 18)

This attitude leads to a conscious and professional treatment of multi-perspectivity between IS experts and organization experts which is more than mere mediation between the traditions of different cultures (ethnographic research) (cf. Järvinen, 2001, pp. 81-87). Our ideas are described in detail in Holl & Feistner (2006).

We understand the cooperation of IS experts and organization experts in the sense of bilateral responsibility and bilateral knowledge gain. Therefore, we would base the cooperative aspect of permanent knowledge gain in organizations upon experts with different backgrounds, upon the dialectic between organization experts and IS experts and their tasks and knowledge.

In the following section, this will become even more obvious when we connect the second aspect of knowledge gain, the cyclic-iterative one, to the mayeutic cycle.

2.2.2 Cyclic-iterative knowledge gain: mayeutic cycle

We now base our argumentation on Section 2.1.2. The term *mayeutic* for the cycle of knowledge gain (of a cognitive process) is an allusion to the *maieutiké téchne* (*μαιευτική τέχνη*, midwifery), a technique Socrates used in combination with his doctrine of *anamnesis* where he tried to show people that they can, just by remembering, reveal unconscious knowledge when they are guided by questions during a dialog.

The commonly better known *hermeneutic cycle* "is essentially a very general model of the development of knowledge" (Radnitzky, 1970, Vol. II, pp. 23-30) as well and contains similarities to a mayeutic cycle. Although hermeneutics is important in IS for the interpretation of the organization experts' statements by IS experts (cf. Section 1.2), we prefer the mayeutic cycle. The latter can be designed in a clearer, more transparent and more elaborate way and cannot be associated with the inherent danger of a hermeneutic circle, that is, that the result of an interpretation can already be part of the premises used.

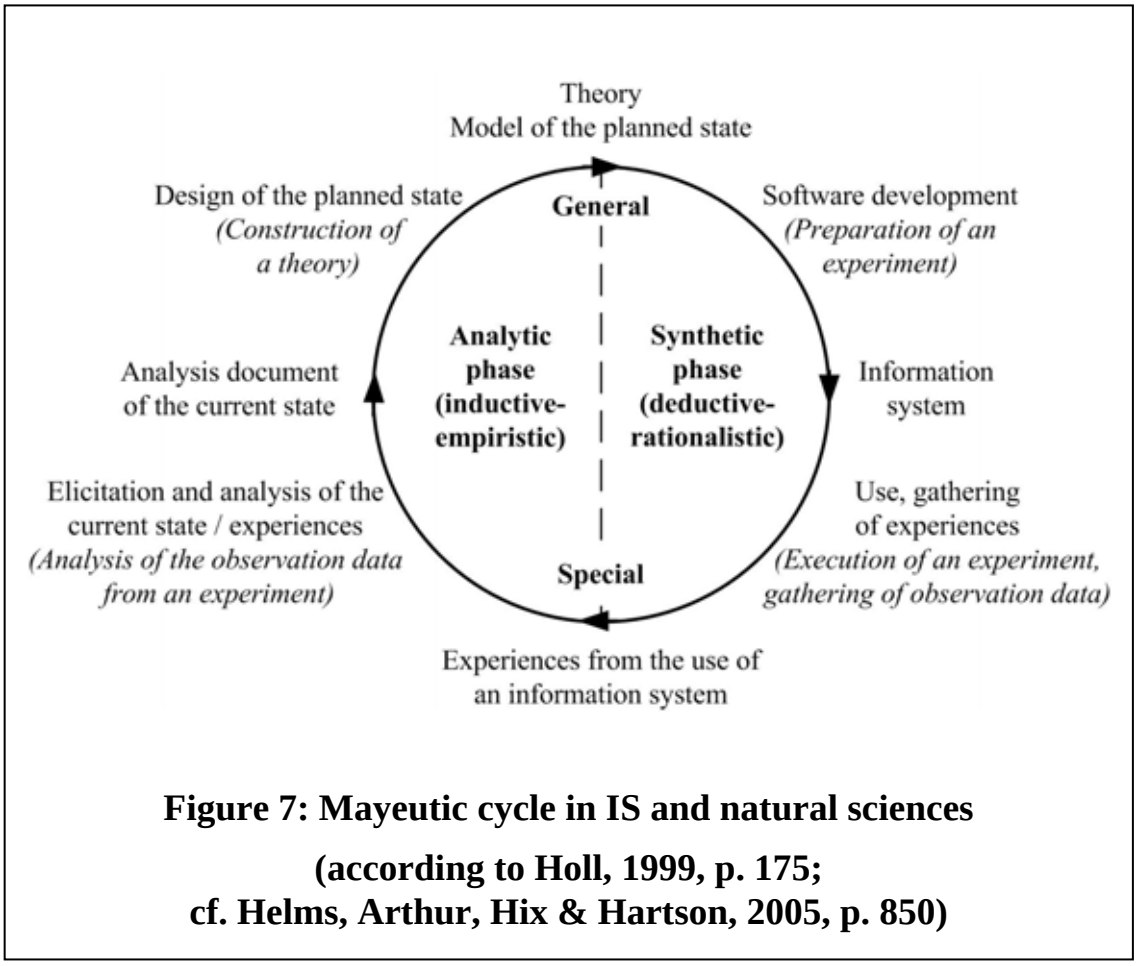


Figure 7: Mayeutic cycle in IS and natural sciences

(according to Holl, 1999, p. 175;
cf. Helms, Arthur, Hix & Hartson, 2005, p. 850)

Therefore, with regard to empiric sciences (the background of the authors), which we consider an adequate paradigm for IS, only the concept of a mayeutic cycle is used in this context.

The mayeutic cycle is a visualization of Kant's unification of empirism and rationalism. It comprises a deductive-rationalistic half (corresponding to the synthetic phase of a software life cycle) and an inductive-empiristic half (corresponding to the analytic phase of a software life cycle). Figure 7 shows the mayeutic cycle in IS (according to the information system life cycle model in Table 1) in comparison to the one in natural sciences (cf. Holl, 1999, p. 175), the latter marked with italicized text in parentheses. The four quadrants represent four phases, the four cardinal points represent the results of the four phases.

The term *mayeutic cycle* (as the one in Figure 7) and its relation to epistemology are widely unknown in IS. The spiral model of software development and enhancement by Boehm (1988), however, is a model approach similar to the mayeutic cycle. The spiral model combines elements of both design and prototyping steps and visualizes the corresponding knowledge gain with an increasing diameter. A simplified version of the spiral model by Sommerville (2001, p. 164) is presented in adapted form in Figure 8.

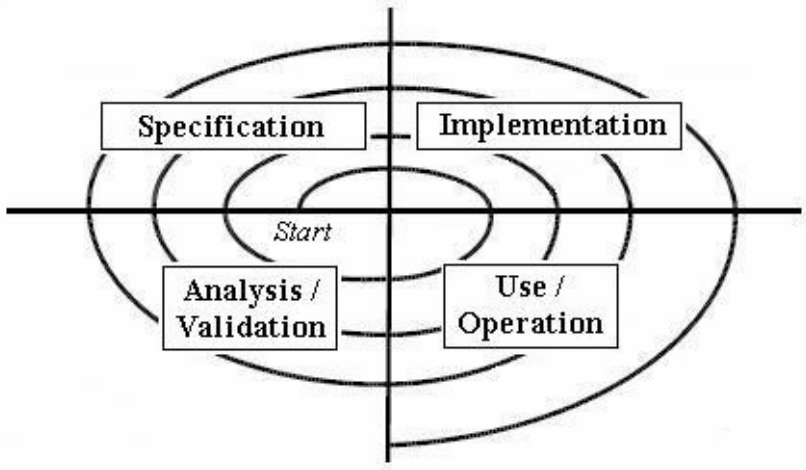


Figure 8: A spiral model of software development and enhancement (adapted from Sommerville, 2001, p. 614 according to Boehm, 1988; cf. Helms, Arthur, Hix & Hartson, 2005, p. 847)

The mayeutic cycle of Figure 7 is now extended to cover the approach of design theory / design research in IS. For this purpose, the mutual connections of IS research to its "environment" (composed of people, business organizations and their technological support) as well as to a "knowledge base" (a pool of foundations and methodologies provided by prior IS research and results from other disciplines) have to be taken into consideration (Figure 9).

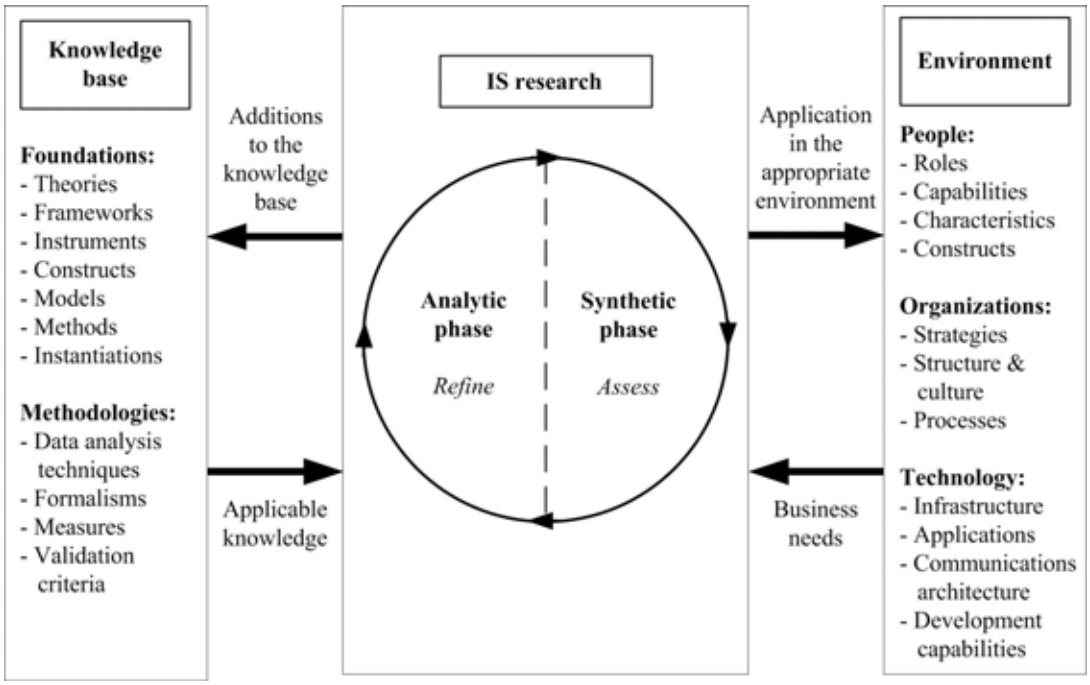


Figure 9: Embedded mayeutic cycle in IS research

(adapted from Hevner, March, Park & Ram, 2004, p. 80)

The cycle of Figure 7 is embedded into two "outer" (also mayeutic) cycles.

Right "outer" cycle: business needs from the environment (defined by goals, tasks, problems or opportunities of people within organizations) are included into IS research which develops theories and models that, in turn, can be applied to appropriate environments.

Left "outer" cycle: applicable knowledge from a knowledge base (theories, models, methods etc.) is included into IS research which, in turn, contributes new or improved theories, models and methods to the knowledge base.

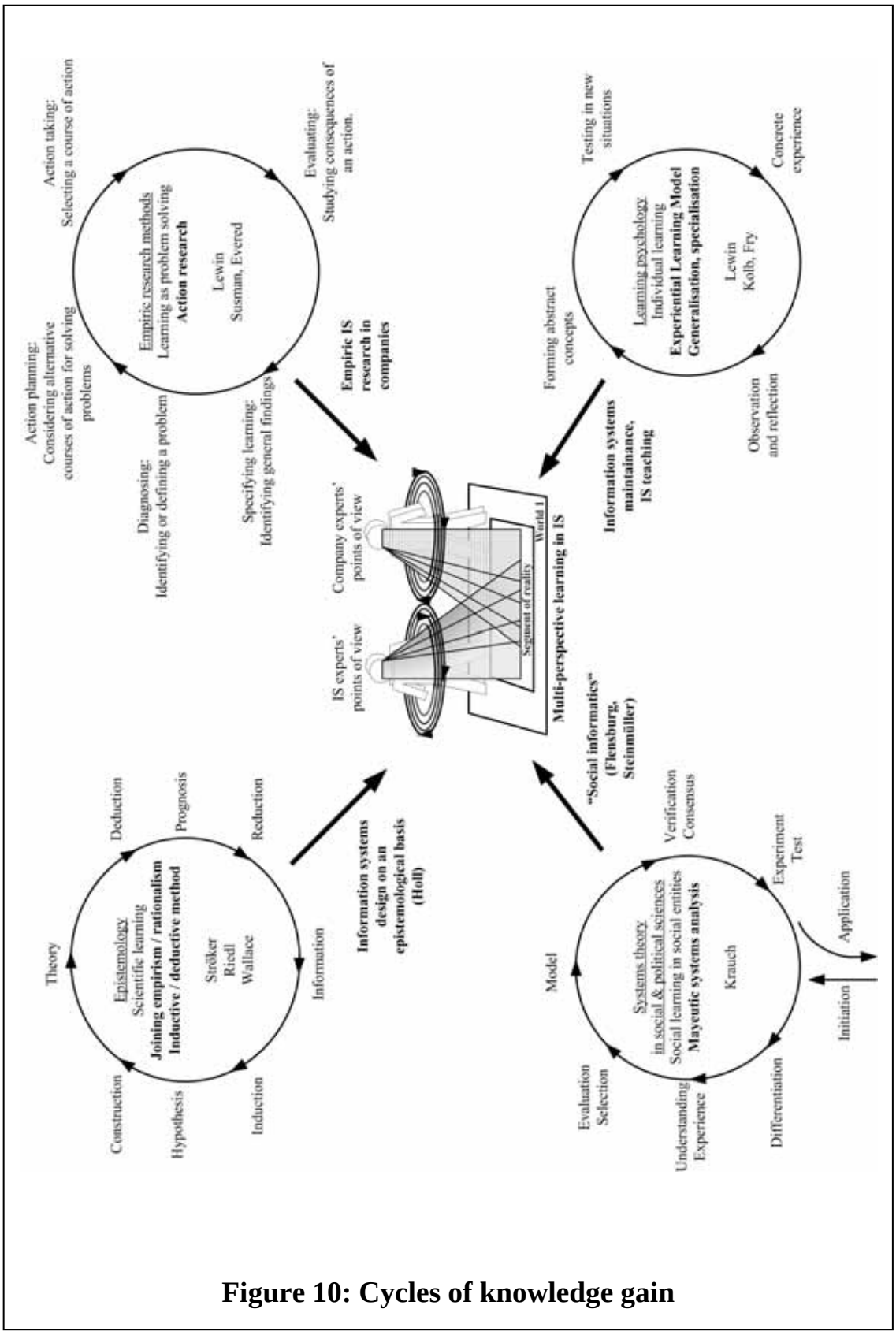


Figure 10: Cycles of knowledge gain

Although not theoretically discussed in IS, mayeutic cycles are well known in other scientific disciplines. In Figure 10, we show four of them which use mayeutic cycles in slightly different forms:

- epistemology: joining empirism and rationalism as well as the inductive and deductive methods (cf. Wallace, 1969 according to Järvinen, 2001, p. 5; Riedl, 1984, p. 186, also verbally underlined by Ströker, 1987, pp. 13-30; Davis & Hersh, 1981, p. 131).

  This cycle is the theoretically most elaborate one which serves as a template for the mayeutic cycle in IS (cf. Figure 7). The other cycles in Figure 10 are equally oriented, with the theory (model) placed at the north pole and the information (experience from the use) at the south pole.

- systems theory in social and political sciences: mayeutic systems analysis (Krauch, 1992, p. 344; cf. Krauch, 1972, p. 41)

- learning psychology: experiential learning model (from http://www.infed.org/biblio/b-explrn.htm, originally from Kolb & Fry, 1975 according to a cyclic spiral model by Lewin, 1948, p. 206)

- empiric research methods: action research (Järvinen, 2001, p. 116 according to Susman & Evered, 1978)

All of the four approaches are correlated with a special type of learning:

- scientific learning (epistemology)
- social learning in social entities, e.g. learning organizations (systems theory in social and political sciences, organization theory)
- individual learning (learning psychology)
- learning as problem solving (empiric research methods)

All of the four approaches are also correlated with CS / IS via:

- epistemology-based information systems modeling (epistemology)
- “socio-informatical” approaches (systems theory in social and political sciences)
- information systems maintenance and IS teaching (learning psychology)
- empiric IS research in organizations (empiric research methods)

As IS experts and organization experts contribute to the permanent knowledge gain in an organization, we can speak of a double cycle of knowledge gain.

The adaptation of Figure 4 in the center of Figure 10 shows IS experts and organization experts who are permanently and iteratively rotating through the different phases of the cycle of knowledge gain explained above.

As the aspect of multi-perspectivity, however, is not represented in detail in a mere double cycle of knowledge gain (in each situation, there are several IS experts and several organization experts), we have to refine this image.

Aggregating the individual cycles of the experts, an image similar to the rings of Saturn can be derived for the group of IS experts as well as for the group of organization experts: from far, one can distinguish two or three, the nearer one gets or the better the magnification becomes, the more rings one can see. Thus, each of the two strings of the double cycle of knowledge gain – regarded from near – turns out to consist of several substrings representing the different individuals on either side.

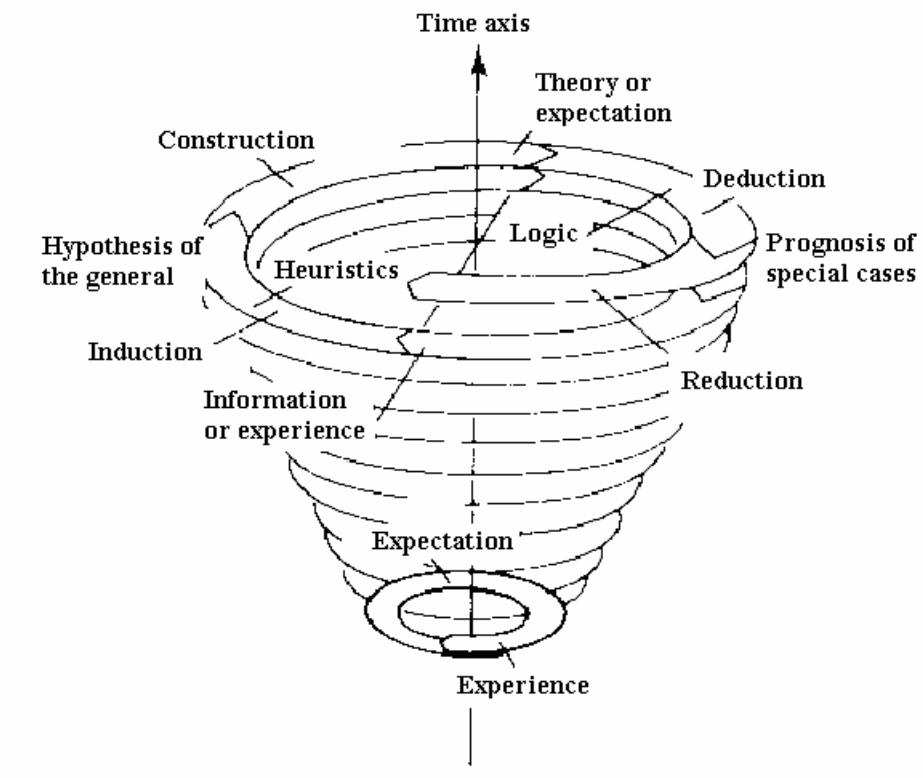


Figure 11: Conic helix of knowledge gain

(adapted from Riedl, 1984, p. 169)

Stretching the cycle of knowledge gain in time, one gets a helix, and if one, in addition, visualizes the increase of knowledge, relative to a specific domain, with an increasing diameter, one arrives at Riedl's conic helix (Figure 11), a visualization used in evolutionary epistemology.

When we give Riedl's helix two strings, representing the knowledge gain – regarding a special problem area – of organization experts and IS experts, we have completed our visualization of cooperative, cyclic-iterative knowledge gain in the title. The same geometrical figure can be obtained by stretching two superposed spirals (cf. Figure 8) in time. Regarded from close, the two strings will of course turn out to consist of several substrings representing the individual knowledge gain of each expert involved.

As IS experts and organization experts suffer from multi-perspectivity in combination with the temporal dynamics of organizations, the question arises of what can be done against the undesired consequences of this special situation. In the following sections, we will deal with one of them, the fast aging of information systems. If no anti-aging methods are applied, the life cycle of information systems will be shortened extremely and, as a result, high costs and great efforts will be required for the design and development of completely new information systems.

3. INFORMATION SYSTEMS ANTI-AGING SUPPORTED BY COOPERATIVE, CYCLIC-ITERATIVE KNOWLEDGE GAIN

3.1 Fundamental definitions

We have defined the term *(technical) information system* as the software part (technical system) of an organizational information system (cf. Section 1.2). We refine this definition to better explain the essential connection between the anti-aging methods – in the different phases of an information life cycle – described in Section 3 and the cooperative and cyclic-iterative aspects of knowledge gain in Section 2. Therefore, we locate information systems in Lehman's classification schema and consider the influences on them. Then we define the term *information systems aging* in the context of this study.

Manny Lehman was one of the first who addressed the topic of information systems aging. In his fundamental article (Lehman, 1980, pp. 1061-1063) he classified three types of IT systems:

- S-type (specifiable): IT systems for problems that can be specified formally and completely. S-type programs typically solve formal mathematical problems. A new requirement leads to a completely new S-type system. S-type systems are not typical for the field of IS.
- P-type (problem solution): IT systems that approximately solve a specific, delimited complex real-world problem, such as a weather forecast. Because of the high complexity of the problem, a solution can never be completely specified. P-type systems do not exert an influence on their surroundings (e.g. collection of weather data cannot directly influence the weather) as information systems do. Therefore, they are not of interest in IS.
- E-type (embedded): IT systems that mechanize a human or societal activity and operate in or address a problem or activity of the real world. An E-type system as a part of its environment re-influences its environment. Lehman described the behavior of E-type systems in the course of time in his "feedback systems" law: it defines E-type systems as multi-level, multi-loop, multi-agent *feedback* systems. E-type systems must be continually changed and updated, i.e. evolved. Information systems are a subgroup of E-type systems. Therefore, we can transfer statements about and experience with E-type systems to information systems.

We will now consider the specific reasons for the aging of information systems. As Lehman explains, E-type systems, in our case information systems, have to be modified constantly during their lives. This is due to technical and business reasons. Technical reasons, such as the change of the organization's IT environment due to performance needs or the migration of an information system due to a merger, are relevant for P-type systems as well. Therefore, we do not consider them in this study and confine ourselves to business reasons which are specific for information systems. The temporal dynamics of an organization (related to the cyclic-iterative aspect of knowledge gain), which inevitably requires changes of its information system, is due to internal and external business reasons.

The internal reasons include all those arising from inside the organization, such as the change of the requirements caused by reflections and experiences with the use of an information system.

The external reasons include all those arising from outside the organization, such as change of laws or change of customer demands which dynamically generate new or changed requirements. They cannot be influenced by the organization, but affect it.

The external and internal reasons lead to changed requirements towards the organization's information system. If the corresponding changes are not done consciously and professionally – this happens often – the quality of the information system decreases. Therefore, the information system gets older, develops "wrinkles" and becomes more difficult to maintain.

The expression "software aging" describing this situation was first introduced by Parnas (1994, p. 280). Parnas distinguishes between:

- *functional aging*: a natural process which has its origin in a lack of adaptation of software to its changed environment ("lack of movement").
- *qualitative aging*: a consequence of non-professional adaptations due to functional aging of software with decay of its internal structure ("ignorant surgery") so that software complexity increases faster than the complexity of its environment.

Qualitative aging is the only interesting type of information systems aging in our context. Therefore, we always mean qualitative aging when we speak of aging and use the term *information systems aging* instead of *software aging*.

We consider it as very important to connect the ideas of Section 2 to the topic of information systems anti-aging:

- to take care for permanent, responsible and mature cooperation of organization experts and IS experts in information systems design (cf. Section 2.2.1)
- to take into consideration the permanent change with the behavior of an organization (supported with an information system) and, therefore, the repetitive cyclic iteration of the maintenance step (consisting of an analytic and a synthetic subphase) (cf. Sections 2.1.2 and 2.2.2)

Therefore, every anti-aging method described in this study will cover the cooperative and / or the cyclic-iterative aspect of knowledge gain. We will confine ourselves to those methods which counteract the specific form of aging of information systems. The anti-aging methods are described in the order of their first appearance in the information system life cycle (cf. Table 1). The first anti-aging method in the life cycle is *requirements engineering* (cf. Section 3.2.1) in combination with local and temporal extrapolation (cf. Section 3.2.2). *Changed requirements management* (cf. Section 3.3.1) already starts within the analytical phase, after requirements engineering, and counteracts information systems aging during the synthetic phase. From our point of view, the essential anti-aging method during the maintenance is *change management* (cf. Section 3.4.3). Figure 12 visualizes the anti-aging methods described in Section 3 and relates them to the information system life cycle phases.

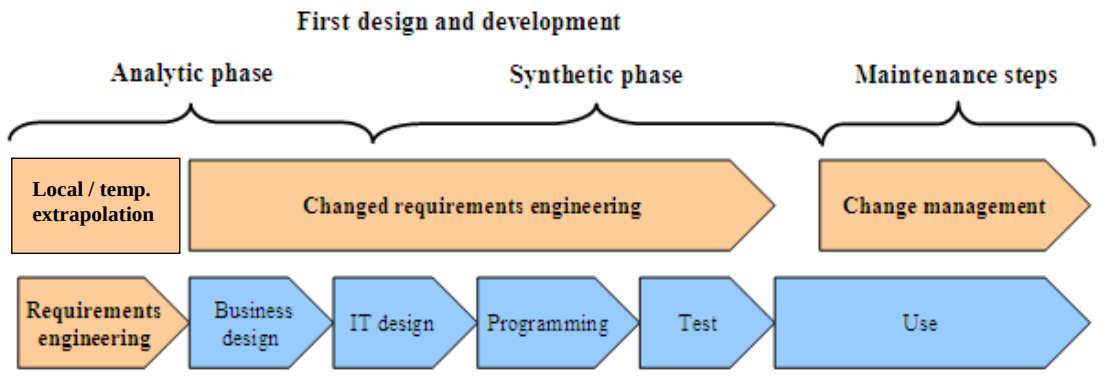


Figure 12: Requirements and information systems life cycle

3.2 Information systems anti-aging methods during the analytic phase

The first phase in an information system development process is the analytic phase (cf. Table 1). Already in this phase, maintainability and changeability as criteria of software quality have to be respected. Failures in requirements definitions often lead to the necessity of a completely new design of entire information systems and thus cause great efforts and costs. As the cooperation of IS experts and organization experts is highly important especially during the analytic phase, it is indispensable to consider it from the very beginning. For instance, the implementation of participatory strategies (Holl, 1999, p. 195, p. 205) can be helpful as well as the improvement of the social and cognitive abilities of both types of experts (cf. Rupp, 2004, p. 94; cf. Section 2.2.1). These methods, however, can only support, but not replace professional requirements engineering.

3.2.1 Requirements engineering

Especially for the design of critical information systems, requirements engineering has become a standard method. It should, however, be used for the development of every information system, which is not done so far. Prototyping is only sufficient and helpful for small information systems and small segments of large information systems. Many IS experts and software developers still have to be taught the advantages of requirements engineering. Therefore, we follow the detailed argumentation in Holl (2004) and Holl & Maydt (2007, pp. 49-51).

Requirements engineering covers, above all, the following partial tasks: requirements acquisition, requirements definition and requirements analysis which is the activity of identifying, formulating and validating the requirements. Furthermore, other tasks, such as the management and adaptation of requirements, are often mentioned. It is not considered a task of requirements engineering to check the observance of the requirements in the course of software development and to provide a general version management with regard to modification (cf. Partsch, 1998, p. 20). As an independent discipline of IS, requirements engineering is a systematic approach to the development of requirements through an iterative process of analyzing the problem, documenting the resulting requirements and checking the accuracy of the understanding so gained (cf. Rzepka 1985, pp. 9-12).

Requirements are statements about the properties and performance of a product, i.e., a requirement describes, among others, a feature to be met by a software product. Requirements are to be systematically acquired, described, analyzed

and completed (cf. Rupp, 2004, p. 11). Since requirements serve as a communication basis for all of the persons involved in the entire software process, they should be described as completely, consistently, comprehensibly, unambiguously and correctly as possible. All of the requirements established during requirements analysis are recorded in requirements documents (cf. Rupp, 2004, pp. 21-24).

Figure 13 shows the general method for the treatment of requirements and, therefore, represents the life cycle of a requirement. The inquiry into requirements has the aim to establish the features which a future information system has to possess. For this purpose, methods, such as interviewing or questioning, are used frequently. After the requirements are acquired, they are recorded in a requirements document. Next, during requirements analysis, the quality of the requirement descriptions is checked by verification and validation. Verification shall find out whether a requirement description meets certain criteria, e.g. completeness. Validation, however, shall detect whether a requirement description represents the customer's wishes adequately. Finally, when accepted, the requirements are handed over to the next step in the software process (cf. Partsch, 1998, pp. 27-37).

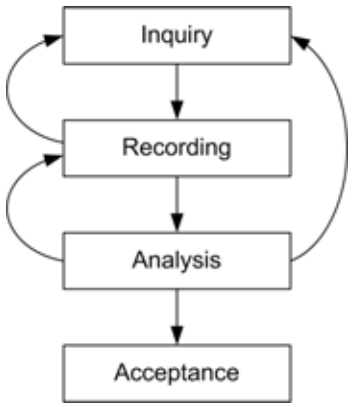


Figure 13: Life cycle of a requirement

(Holl & Maydt, 2007, p. 49; adapted from Partsch, 1998, p. 27)

The main focus of requirements engineering is the mature cooperation of IS experts and organization experts, i.e., the cooperative aspect of knowledge gain. An advanced form of requirements engineering also considers possible future changes, i.e., the cyclic-iterative aspect of knowledge gain (cf. Sections 3.2.2 and 3.3.1).

3.2.2 Open models: local and temporal extrapolation

According to the experience of one of us (AH), there are two advanced methods that should also be considered during the analytic phase before the final design (and further on during the analytic subphases of the maintenance steps) in order to keep the design flexible:

- *"Local" extrapolation* regarding extensions

  Do any other business activities already exist which should or could later become the subject of an extended information system? There is no need to program an IT support for these areas from the very beginning, but one should include them in the information system design.

- *"Temporal" extrapolation* regarding modifications

  Are there any expected changes in the business activities? This information should also be considered in the information system design.

To extrapolate means "to project, extend or expand known data or experience into an area not known or experienced so as to arrive at usually conjectural knowledge of the unknown area"; extrapolations are not restricted to numeric ones.

An example for *local extrapolation* is the database of a course information system of a CS department at a university. In the database, information is stored about lecturers and courses. Every course is given by exactly one lecturer. Therefore, only two database tables are needed: one containing data about the lecturers with an artificial Lecturer_ID as primary key and another one containing data about courses with an artificial Course_ID as primary key and the Lecturer_ID as foreign key (Figure 14).

Figure 14: Data model before local extrapolation

This database design is absolutely correct regarding the current situation in the CS department (one course is always given by exactly one lecturer). Let us assume that the IS department already offers courses given by more than one

lecturer. A future extension to this department, however, will lead to serious problems. In this case, a structural change of the database design becomes necessary: the introduction of a third table assigning lecturers to courses and using a combined primary key (Course_ID + Lecturer_ID) (Figure 15).

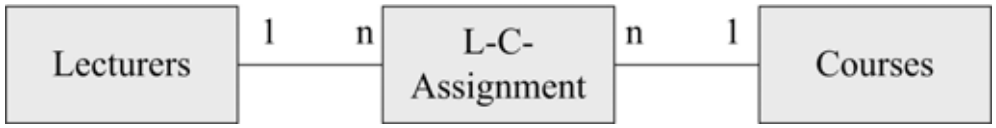


Figure 15: Data model after local extrapolation

The new model is more flexible, but completely meets the original requirements as well: if a course is given by only one lecturer, it has only one entry in the assignment table.

Structural changes of the underlying data model of an information system lead to extra efforts as they require single-use programs to move the data from the old data structure to the new one. Therefore, structural changes have to be counteracted using flexible models which apply to general situations rather than to the current special ones only. The problem mentioned can be avoided with early local extrapolation.

Even if the information system remains confined to the CS department, there can be changes as well. If a future development in the CS department itself leads to courses given by more than one lecturer, the same example illustrates *temporal extrapolation*.

A well-known example for temporal extrapolation is the so-called millennium bug (also known as year-2000-problem / Y2K-problem). In the 20th century, some computer programs stored year data only in a two digit format. After the turn from 1999 to 2000, these programs where not able to distinguish between the centuries, e.g. “1900” was stored as “00” as well as 2000. The consequences were errors in calculations, for example with regard to a person’s age by subtracting the person’s year of birth from the current year. In the year 2000 (interpreted as “00”), the calculated age of a person born in 1968 (stored as “68”) was negative (00 - 68 = -68) (Figure 16). An early temporal extrapolation with the consequence of storing year data in a four digit format would have completely avoided this problem.

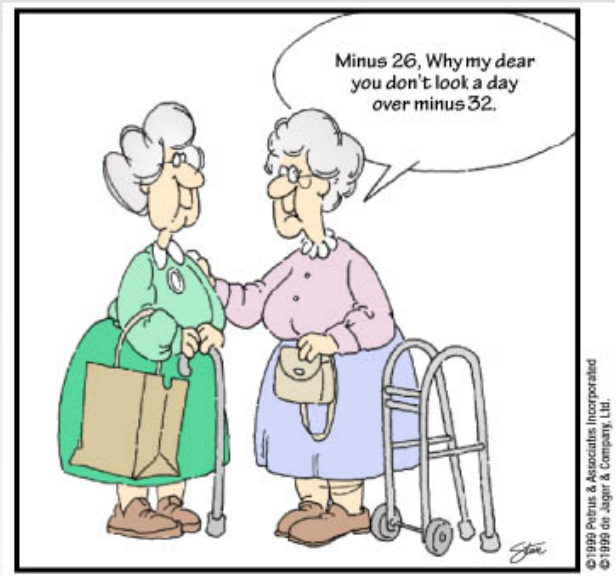


Figure 16: Database millennium bug

(cf. http://mlecture.uni-bremen.de/intern/ws2005_2006/fb03/vak-03-706.1/20051024/folien.pdf)

Extrapolations regarding extensions and modifications keep information system designs more flexible and future-oriented (the cyclic-iterative aspect of knowledge gain). Contrary to the common situation of passive reactive designs which do not change before a change in the organization has happened, we, therefore, support the idea of an *active design.* It tries to anticipate and include future changes in an organization from the very beginning. We call the resulting models *open models*.

Information about expected extensions and modifications, however, can only be acquired by responsible cooperation of organization experts and IS experts (the cooperative aspect of knowledge gain).

Advanced requirements engineering and extrapolation help to prevent information systems aging already within the analytic phase during the first design. Each of these two methods covers either of the two aspects of the "cooperative cycle" and helps to reduce permanent corrections in the future, which in consequence counteracts information systems aging.

In order to guarantee the success of a project and to avoid early aging, the application of adequate methods has to be continued during the synthetic phase.

3.3 Information systems anti-aging methods during the synthetic phase

3.3.1 Changed (creeping) requirements management

A method that should be applied during the synthetic phase is changed requirements management. It has to counteract a gap between the current state of an organization and an information system at the time of its deployment. Such a gap is due to changes in the organization's behavior that appear during software development after the end of requirements engineering.

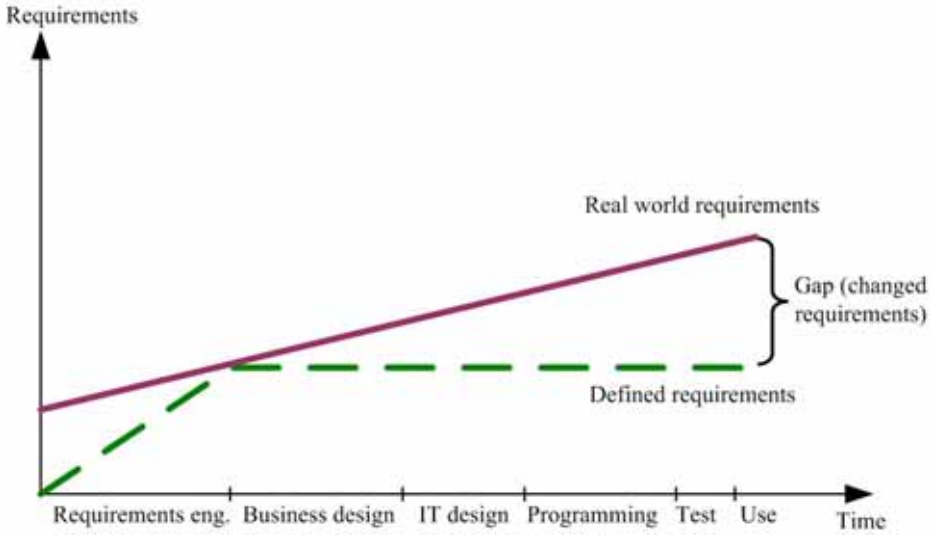


Figure 17: Changed requirements gap

Figure 17 visualizes this situation. The phases of the software life cycle are shown on the time-axis while the requirements appear on the ordinate. The dashed line represents the requirements defined during the analytic phase. Usually, IS experts do not change them during software development after requirements engineering has been finished. The continuous line (used as simplifying visualization for a chaotic process in the real world) represents the changes of the organization which correspond to changes of the real-world requirements. The gap between the dashed and the continuous line visualizes the difference between the real world and the defined requirements. If the changing requirements (also known as *creeping requirements*) are not taken into consideration during the synthetic phase, the information system is already out of date at the time of its deployment.

A possible method for minimizing the changed requirements gap is *continuous requirements approval*: after each development step the organization experts are asked to reconfirm their requirements. Thus, a mutual cooperation of IS experts and organization experts is essential for managing changed requirements, i.e., the cooperative aspect of knowledge gain. The dynamics of the real-world requirements (which results in creeping requirements) corresponds to the cyclic-iterative aspect of knowledge gain.

Further methods for the synthetic phase will not be discussed in detail in this study as they are not specific for information systems. Among others, there are the use of coding conventions (on module and line level), prototyping, design patterns, unit tests, reviews and, very often neglected, the documentation of IT design and code (Figure 18).

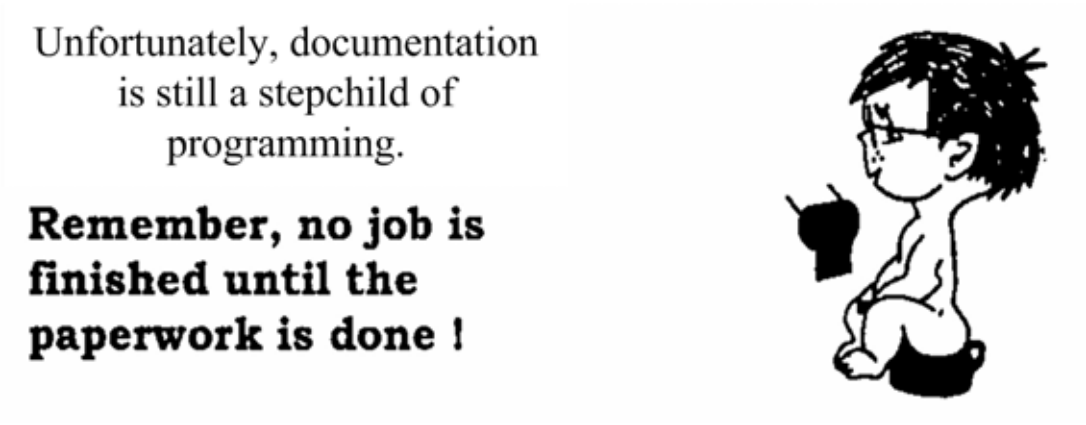


Figure 18: Paperwork

(adapted from Dreehsen, 1996, p. 80)

3.4 Information systems anti-aging methods during maintenance

The huge part of work during maintenance in an information system life cycle is illustrated in an excellent way in Figure 19.

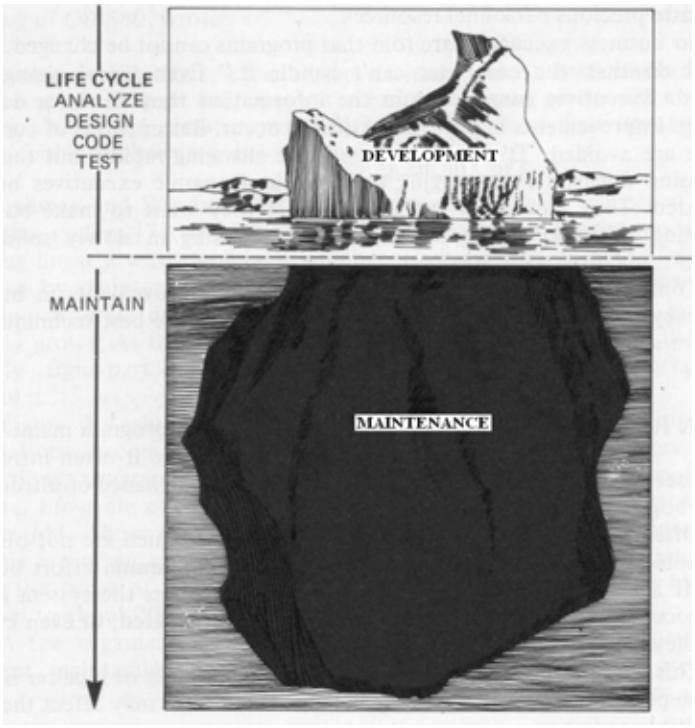


Figure 19: The iceberg of software maintenance

(Martin & McClure, 1983, p. 7)

The tip of the iceberg (information systems development) is well perceivable whereas information systems maintenance is hidden under the surface of the ocean. The maintenance of today's information systems needs a tremendous effort. Time and costs do not only often exceed the investment of the development, but are also unpredictable, difficult to plan and include incalculable surprises. Therefore, implementing and managing change of in-house and customizable off-the-shelf information systems is a key problem for organizations. Maintenance, however, is inevitable to keep information systems up to date. For this reason, one should be familiar with the possible maintenance types and the corresponding anti-aging methods specific for information systems.

3.4.1 Types of information systems maintenance

According to our personal business experiences, we distinguish between different maintenance activities. To classify them, we adopt Swanson's three types of software maintenance (Swanson, 1976, pp. 492-497) and a fourth type from Kroha (1997, p. 181) and refine their definitions. Each information systems maintenance type can be split into a technical and a business subtype (cf. Section 3.1). Each subtype will be judged with regard to its importance for this study.

- *corrective*:
    - technical: maintenance performed to correct faults in hardware or software, also known as *issue management* and *bug fixing.*
    - business: maintenance performed to correct incorrectly implemented requirements or to implement defined but not implemented requirements.

  These types are necessary for every type of IT system (e.g. P-type) and not specific for information systems. Therefore, they are not in the scope of this study.
- *adaptive*:
    - technical: maintenance performed to render an information system usable in a changed technical environment (e.g. in the environment of a new operation system) or to add new technical functions (e.g. multi-user functionality). This type is necessary for every IT system (e.g. P-type), therefore, it is not in the focus of this study.
    - business: maintenance performed to render an information system usable in a changed business environment (e.g. changed organization structure) and to meet the corresponding new business requirements (e.g. customer demands). This type is specific for information systems. A method of adaptive maintenance is *change management* (cf. Section 3.4.3).
- *perfective*:
    - technical: maintenance performed to improve immanent properties of an information system, such as performance and maintainability, without adding new functionality. A method of perfective maintenance is, among others, *reengineering* (cf. Section 3.4.4). Like every technical-perfective maintenance method, reengineering is not specific for information systems. Due to its tremendous practical relevance for already aged information systems, we, however, will take it into consideration in this study.

- business: maintenance performed to adjust and "polish" the requirements. This type is not necessary if requirements engineering is done professionally (cf. Section 3.2.1). Therefore, it is not interesting in the context of this study.

- *preventive*:
  - technical: maintenance performed to facilitate future technical-adaptive maintenance activities, such as keeping the hardware requirements of the information system flexible and independent. This type of maintenance is reasonable for every type of system (e.g. P-type) and not specific for information systems. Therefore, we do not consider it in this study.
  - business: maintenance performed to facilitate future business-adaptive maintenance activities. This type of maintenance is specific for information systems. Possible methods for business-preventive maintenance are *local* and *temporal extrapolation* which have already been explained (cf. Section 3.2.2).

Regarding these definitions, the maintenance types relevant in the scope of this study are technical-perfective, business-adaptive and business-preventive maintenance. Each maintenance type requires appropriate anti-aging methods. The selection in this study is based on the fundamental property that each method, except for reengineering, can be described with a mayeutic cycle regarding the cyclic-iterative aspect of knowledge gain.

Before we describe the anti-aging methods mentioned above, we consider the impact of technical-perfective and business-adaptive maintenance on information systems aging. For this purpose, we regard two of Lehman's laws of software evolution.

3.4.2 Types of information systems maintenance and their relation to Lehman's laws of software evolution

Lehman's observations in the field of software evolution over a period of 30 years led to his definition of eight laws of software evolution. These laws are specifically related to E-type system evolution. Therefore, we can apply them to information systems. Two of these laws describe the growth of complexity depending on the changes in an organization.

- law of *continuing growth:* the functional capability of E-type systems must be continuously increased to maintain user satisfaction over the system lifetime (Lehman & Belady, 1972).

- law of *increasing complexity:* as an E-type system evolves, its complexity increases unless work is done to maintain or reduce it (Lehman & Belady, 1972)

Figure 20 visualizes the relation between the two laws and information systems aging.

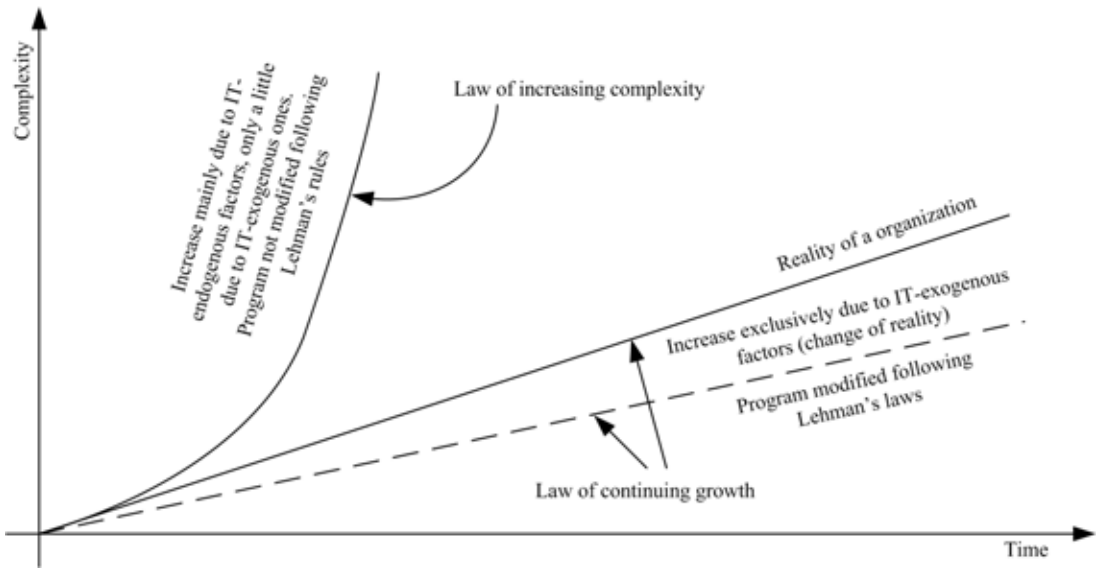


Figure 20: Increasing complexity of E-type systems over time

Due to the temporal dynamics of organizations, it is inevitable to keep information systems up to date. This means that new functions have to be added or old ones have to be adapted throughout the entire lifetime of an information system. These changes, in turn, increase the information system's complexity and lead to its aging. Maintenance methods have to be applied to counteract this process of aging. If maintenance methods are not applied or if they are not applied professionally, the complexity of the information system will increase disproportionately. The other way round, if they are applied professionally, the complexity of the information system will grow less fast than the complexity of the real world. These considerations apply especially to business-adaptive maintenance.

3.4.3 Change management

According to our business experiences, business-adaptive maintenance, such as adjustments or the addition of new functionality to an information system, are often done in a quick and dirty way: the implementation of new functions is not based on a formalized model of the information system and its business environment. This means that some or all of the steps between the experiences from the use of an information system and the software development in a mayeutic cycle (cf. Figure 7) are not performed professionally and completely or are even skipped. This will increase an information system's complexity and, therefore, lead to its aging. Only if maintenance is done professionally, i.e., in a full mayeutic cycle, it will counteract the information system's aging.

A method of performing professional business-adaptive maintenance is called *change management*. Change management is an excellent example for a full mayeutic cycle (cyclic-iterative aspect of knowledge gain) which is repeated in several iterations. We suppose that one cycle consists of two subphases, an analytic one and a synthetic one, similar to the corresponding phases of an information system life cycle (cf. Table 1).

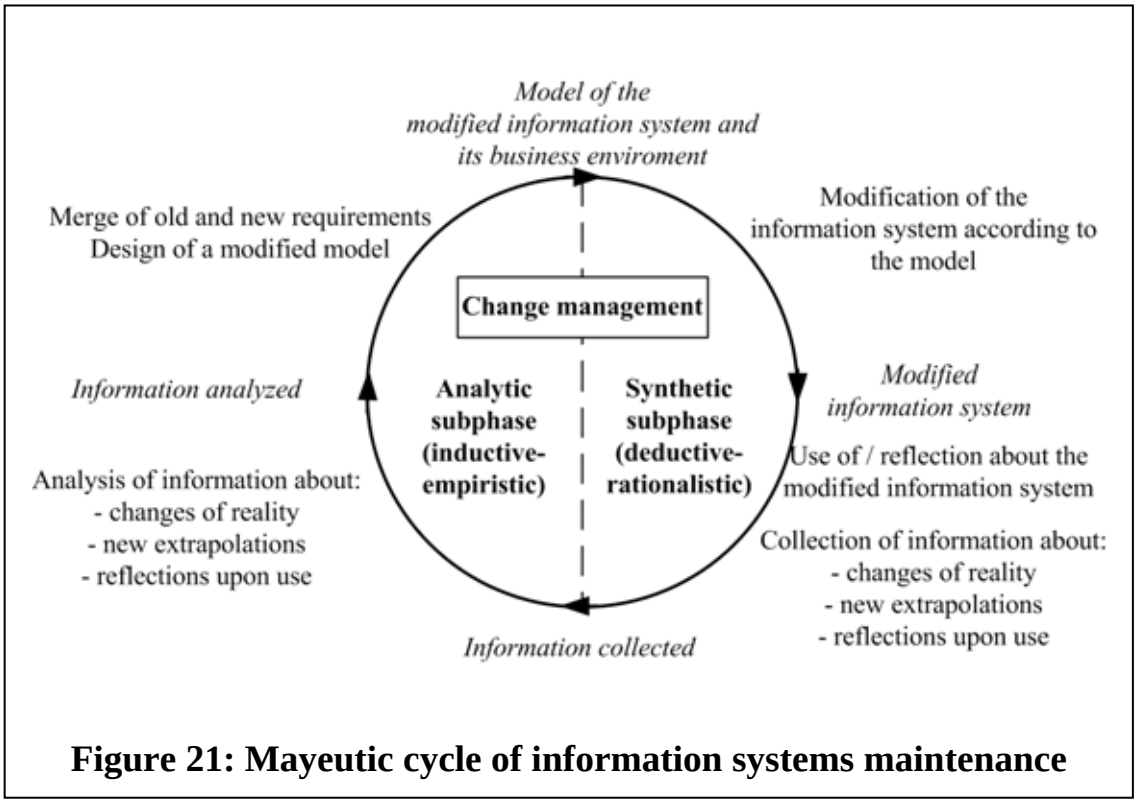


Figure 21: Mayeutic cycle of information systems maintenance

The analytic subphase corresponds to the inductive half of a mayeutic cycle, the synthetic one to the deductive half. The synthetic subphase will always collect changes of reality, reflections upon use and new extrapolations. The analytic subphase will analyze them and create a new model of the planned state which will be transformed into a modified information system in the synthetic phase (Figure 21).

Therefore, the methods of Section 3.2 also apply for the analytic subphase of a change management cycle as well as the methods of Section 3.3 do for its synthetic subphase. All of the methods for business-adaptive maintenance require a close and responsible cooperation of IS experts and organization experts (cooperative aspect of knowledge gain).

According to our business experiences, change management is not being applied as a standard maintenance method up until now. Like requirements engineering, many IS experts and software developers still have to be taught the advantages of change management. Therefore, there are lots of already aged information systems which have to be dealt with using methods of technical-perfective maintenance, such as reengineering.

3.4.4 Information systems reengineering

Reengineering, as a method for technical-perfective maintenance, can serve as a practical solution to reduce the complexity of already aged information systems. The immanent technical properties of information systems will be reorganized and modified in order to make information systems better maintainable or to increase their performance. In contrast to the development of a completely new information system, reengineering does not start from scratch. That leads to a reduced risk of losing knowledge from the information system and to reduced costs of development (Sommerville, 2001, p. 632).

Reengineering is the combination of reverse engineering, followed by forward engineering (Chikofsky & Cross, 1990, p. 15). Figure 22 visualizes the reengineering process. Reverse engineering can be interpreted as the analytic half and forward engineering as the synthetic half of a mayeutic cycle.

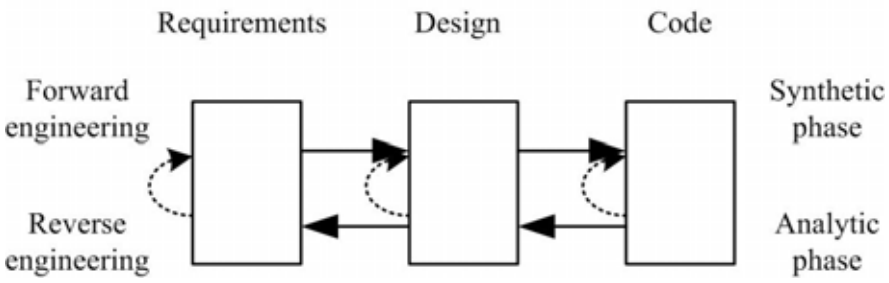


Figure 22: Reengineering process

(adapted from http://mlecture.uni-bremen.de/intern/ws2005_2006/fb03/vak-03-706.1/20051024/folien.pdf according to Chikofsky & Cross II, 1990)

Reverse engineering can be defined as the process of analyzing an information system to identify its components and their interrelationships and to create representations of the information system at higher levels of abstraction. Reverse engineering is a process of examination, not a process of change and replication (Chikofsky & Cross, 1990, p. 15). It can support maintenance by reconstructing an IT design that is unknown or lost. Reconstructed requirements can be used as an input to change management in order to modify an information system (cf. Sommerville, 2001, pp. 636-637).

Forward engineering is the traditional process of moving from high-level abstractions and logical, information-relevant models to the physical implementation of an information system (Chikofsky & Cross, 1990, p. 14; cf. Figure 22).

Although reengineering changes an information system, it does not include new requirements nor new features and functionalities. Reengineering is usually done by IS experts without any influence of organization experts. It, therefore, corresponds only to the cyclic-iterative aspect of knowledge gain. Due to these limitations, one should prefer a professional maintenance method such as change management that consists of a full mayeutic cycle, instead of a method that only repairs the undesired consequences of earlier mistakes and non-professional work.

4. CONCLUSION

With regard to information systems modeling, we have underlined two main features of organizations: their temporal dynamics (cyclic-iterative aspect of knowledge gain) and their behavior as social systems where observation is accompanied by mutual influence of observer and *observandum* (cooperative aspect of knowledge gain), resulting in multi-perspective observation. Based upon these two aspects, we have developed our interpretation of permanent knowledge gain using the concept of mayeutic cycles. A conic double helix turns out to be an excellent visualization for professional information systems maintenance.

Our focus is not software maintenance in general, but information systems maintenance in particular, with emphasis on professional adaptive maintenance.

The anti-aging methods discussed have been presented with respect to our interpretation of permanent knowledge gain (containing the two aspects multi-perspectivity and temporal dynamics) as each of the selected methods is connected to at least one of the two aspects mentioned.

In order to avoid information systems aging, information systems have to be developed and maintained in a professional way. Therefore, essential prerequisites are, among others:

- application of methods representing full mayeutic cycles including an analytic phase (e.g. change management, requirements engineering)
- application of methods requiring and encouraging the responsible and effective cooperation of IS experts and organization experts (e.g. change management, requirements engineering, local and temporal extrapolation, changed requirements management)

Some of the methods introduced (as requirements engineering, local and temporal extrapolation or changed requirements management) are not well known in IS theory so far, others are interpreted in a new way (e.g. change management as the sum of all activities of professional business-adaptive maintenance, or information systems reengineering as an auxiliary technique in case of inadequate change management).

5. REFERENCES

Bhaskar, R. (1989). *Reclaiming reality*. London: Verso.

Boehm, B. W. (1988). A spiral model of software development and enhancement, *IEEE Computer*, 21(5), 61-72.

Bülow, V. von (1968). *Loriots großer Ratgeber*. Zürich: Diogenes.

Castelli, V., Harper, R. E., Heidelberger, P., Hunter, S. W., Trivedi, K. S., Vaidyanathan, K. et al. (2001). Proactive management of software aging. *IBM Journal of Research and Development*, 45(2), 311-332.

Checkland, P. (1981). *Systems thinking, systems practice*. Chichester: John Wiley and Sons.

Chikofsky, E. J. and Cross II, J. H. (1990). Reverse engineering and design recovery. A taxonomy. *IEEE Software*, 7(1), 13-17.

Churchman, C. W. (1979). *The system approach and its enemies*. New York: Basic Books.

Churchman, C. W. (1971). *The design of inquiring systems*. New York: Basic Books.

Curth, M. A. and Giebel, M. L. (1989). *Management der Software-Wartung*. Stuttgart: Teubner.

Davis, P. J. and Hersh, R. (1981). *The mathematical experience*. Boston, Mass.: Birkhäuser.

Dreehsen, B. (1996). *Qualitätssicherung bei EDV-Systemen. Auswahl, Einsatz und Betrieb von Hard- und Software gemäß DIN/ISO 9000*. Berlin: Springer.

Eisner, P. (1988). *Strukturierte Software-Wartung*. Zürich: Zentralstelle der Studentenschaft. (Doctoral dissertation, University of Zürich, 1987).

Flensburg, P. and Friis, S. (1999). *Mänskligare datasystem: Utveckling, användning och principer*. Lund: Studentlitteratur.

Goorhuis, H. (1994). *Konstruktivistische Modellbildung in der Informatik*. (Doctoral dissertation, University of Zürich, 1994).

Hajos, A. (1991). *Einführung in die Wahrnehmungspsychologie*. Darmstadt: Wissenschaftliche Buchgesellschaft.

Heidegger, M. (1962). *Being and time*. New York: State University of New York.

Heinemann, K. (1987). *Software-Wartung. Ein modellgestützter Ansatz zur Planung von Software-Wartungsstrategien*. Münster: Lit.

Helms, J. W., Arthur, J. D., Hix, D. and Hartson, H. R. (2006). A field study of the wheel – a usability engineering process model. *Journal of Systems and Software*, 79, 841-858.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75-105.

Hirschheim, R. and Klein, H. K. (2003). Crisis IS field? A critical reflection on the state of the discipline. *Journal of the Association for Information Systems (JAIS)*, 10(4), 237-293.

Holl, A. and Maydt, D. (2007). Epistemological foundations of requirements engineering. In Erkollar, A. (Ed.), *Enterprise and business management. A handbook for educators, consulters and practitioners* (pp. 31-58). Marburg: Tectum.

Holl, A. and Feistner, E. (2006). *Mono-perspective views of multi-perspectivity: Information systems modeling and 'The blind men and the elephant'*. Växjö: Växjö University Press.

Holl, A. (2004). Erkenntnistheorie, (Wirtschafts-)Informatik und Requirements Engineering. In Rupp, C., *Requirements-Engineering und -Management – Professionelle, iterative Anforderungsanalyse für die Praxis* (3rd ed.) (pp. 13-15). München: Hanser.

Holl, A. and Auerochs, R. (2004). Analogisches Denken als Erkenntnisstrategie zur Modellbildung in der Wirtschaftsinformatik. In Frank, U. (Ed.), *Wissenschaftstheorie in Ökonomie und Wirtschaftsinformatik. Theoriebildung und -bewertung, Ontologien, Wissensmanagement* (pp. 367-389). Wiesbaden: DUV.

Holl, A. and Valentin, G. (2004). Structured business process modeling. *Information Systems Research in Scandinavia (IRIS'27)*, CD-ROM. Falkenberg, Sweden.

Holl, A. and Krach, T. (2002). Ubiquitäre IT – ubiquitärer naiver Realismus. In Britzelmaier, B. et al. (Eds.), *Der Mensch im Netz. Ubiquitous Computing. - 4. Liechtensteinisches Wirtschaftsinformatik-Symposium an der FH Liechtenstein* (pp. 53-69). Stuttgart: Teubner.

Holl, A., Krach, T. and Mnich, R. (2000). Geschäftsprozessmodellierung und Gestalttheorie. In Britzelmaier, B. et al. (Eds.), *Information als Erfolgsfaktor. 2. Liechtensteinisches Wirtschaftsinformatik-Symposium an der FH Liechtenstein* (pp. 197-209). Stuttgart: Teubner.

Holl, A. (1999). Empirische Wirtschaftsinformatik und evolutionäre Erkenntnistheorie. In Becker, J. et al. (Eds.), *Wirtschaftsinformatik und Wissenschaftstheorie. Bestandsaufnahme und Perspektiven* (pp. 163-207). Wiesbaden: Gabler [English version 'Information systems as empirical science

and evolutionary epistemology' on Alfred Holl's homepage via http://www.ohm-hochschule.de].

Holl, A. and Scholz, M. (1999). Objektorientierung und Poppers Drei-Welten-Modell als Theoriekerne der Wirtschaftsinformatik. In Schütte, R. et al. (Eds.), *Wirtschaftsinformatik und Wissenschaftstheorie. Grundpositionen und Theoriekerne. Arbeitsbericht 4 des Instituts für Produktion und industrielles Informationsmanagement* (pp. 91-105, 168-169). Essen: Universität.

Järvinen, P. (2001). *On research methods*. Tampere: Tampereen Yliopistopaino.

Jones, C. (1996). Strategies for managing requirements creep. *IEEE Computer*, 29(6), 92-94.

Klein, H. K. and Myers, M. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1), 67-97.

Kolb. D. A. and Fry, R. (1975). Towards an applied theory of experiential learning. In Cooper, C. L. (Ed.), *Theories of group processes*. London: John Wiley.

Krauch, H. (1992). Systemanalyse. In Seiffert, S. and Radnitzky, G. (Eds.), *Handlexikon zur Wissenschaftstheorie* (pp. 338-344). München: Deutscher Taschenbuch-Verlag.

Krauch, H. (1972). Wege und Aufgaben der Systemforschung. In Krauch, H. (Ed.), *Systemanalyse in Regierung und Verwaltung* (pp. 27-47). Freiburg: Rombach.

Kroha, P. (1997). *Softwaretechnologie*. München: Prentice Hall.

Lehman, M. M. (1980). Programs, life cycles and laws of software evolution. *Proceedings of the IEEE*, 68(9), 1060-1076.

Lehman, M. M. and Belady, L. (1972). An introduction to program growth dynamics, statistical computer performance evaluation. In Freiburger, W. (Ed.), *Statistical computer performance evaluation* (pp. 503-511). New York: Academic Press.

Lehman, M. M. (1969). *The programming process*. Yorktown Heights, NY: IBM Res. Rep. RC 2722.

Lehner, F. (1991). *Software-Wartung. Management, Organisation und methodische Unterstützung*. München: Hanser.

Lehner, F. (1989). *Nutzung und Wartung von Software. Das Anwendungssystem-Management*. München: Hanser.

Lewin, K. (1948). *Resolving social conflicts: Selected papers on group dynamics*. New York: Harper and Row.

Lorenz, K. (1978). *Behind the mirror – A search for a natural history of human knowledge*. New York: Harcourt Brace Jovanovich.

Lorenz, K. (1962). Kant's doctrine of the a priori in the light of contemporary biology. *General systems*, 7, 23-35 (= Kant's doctrine of the a priori in the light of contemporary biology. In Bertalanffy, L. von and Rapoport, A. (Eds.), *Yearbook of the Society for General Systems Research* (pp. 23-35)) (Reprinted in Evans, R. (1975), *Konrad Lorenz: The man and his ideas* (pp. 181-217). Reprinted in Plotkin, H. (1982), *Learning, development and culture* (pp. 121-143).)

Lorenz, K. (1962). Gestalt perception as fundamental to scientific knowledge. *General systems*, 7, 37-56 (= Gestalt perception as fundamental to scientific knowledge. In Bertalanffy, L. von and Rapoport, A. (Eds.), *Yearbook of the Society for General Systems Research* (pp. 37-56).)

Martin, J. and McClure, C. (1983). *Software maintenance: The problem and its solution*. Englewood Cliffs, New Jersey: Prentice-Hall.

Nissen, Hans Erik (2002). Challenging traditions of inquiry in software practice. In Dittrich, Y. et al. (Eds.), *Social thinking – software practice* (pp. 69-89). Cambridge, Mass.: MIT Press.

Parnas, D. L. (1994). Software aging. *Proceedings of the 16th International Conference on Software Engineering (ICSE)* (pp. 279-287). Los Alamitos, CA: IEEE Computer Society Press.

Partsch, H. (1998). *Requirements Engineering systematisch – Modellierung für softwaregestützte Systeme*. Berlin: Springer.

Popper, K. R. and Eccles, J. C. (1981). *The self and its brain*. New York: Springer International.

Popper, K. R. (1972). *Objective knowledge – An evolutionary approach*. Oxford: Clarendon.

Radnitzky, G. (1970). *Contemporary schools of meta-science* (2nd ed.). New York: Humanities Press and Gothenburg: Akademiförlaget

Riedl, R. (1984). *Biology of knowledge. The evolutionary basis of reason*. Chichester: Wiley.

Rupp, C. (2004). *Requirements-Engineering und -Management – Professionelle, iterative Anforderungsanalyse für die Praxis* (3rd ed.). München: Hanser.

Russell, B. (1927). *An outline of philosophy*. London: George Allen and Unwin.

Rzepka, W. E. and Ohno, Y. (1985). Requirements engineering requirements: Software tools for modeling user needs. *IEEE Computer (special issue)*, 18(4), 9-12.

Schmidt, B. (1993). *Simulation in Passau*, 2, 12.

Schmidt, H. (1982). *Philosophisches Wörterbuch* (21st ed.). Stuttgart: Kröner.

Sneed, H. M. (1992). *Softwarewartung und -wiederverwendung. Bd. II: Softwaresanierung (Reverse und Reengineering)*. Köln: Rudolf Müller.

Sneed, H. M. (1991). *Softwarewartung und -wiederverwendung. Bd. I: Softwarewartung*. Köln: Rudolf Müller.

Sommerville, I. (2001). *Software engineering* (6th ed.). München: Pearson Studium.

Steinmüller, W. (1993). *Informationstechnologie und Gesellschaft: Einführung in die angewandte Informatik*. Darmstadt: Wissenschaftliche Buchgesellschaft.

Ströker, E. (1987). *Einführung in die Wissenschaftstheorie* (3rd ed.). Darmstadt: Wissenschaftliche Buchgesellschaft.

Susman, G. I. and Evered, R. D. (1978). An assessment of the merits of scientific action research. *Administrative Science Quarterly*, 23(4), 583-603.

Swanson, E. B. (1976). The dimensions of maintenance. *Proceedings of the IEEE/ACM Second International Conference on Software Engineering* (pp. 492-497).

Ulrich, H. and Probst, G. J. B. (Eds.) (1984). *Self-organization and management of social systems: insights, promises, doubts and questions*. Berlin: Springer.

Vollmer, G. (1992). Evolution and projection – Approaches to a modern epistemology. *Universitas* 34(2), 114-126.

Vollmer, G. (1990). *Evolutionäre Erkenntnistheorie* (5th ed.). Stuttgart: Hirzel.

Vollmer, G. (1987). What evolutionary epistemology is not. In Callebaut, W. and Pinxten, R. (Eds.), *Evolutionary epistemology – A multiparadigm program* (pp. 203-221). Dordrecht: Reidel.

Wallace, W. L. (1969). *Sociological theory*. Chicago: Aldine.

BIOGRAPHIES

Alfred Holl

Born 1956. Studies of mathematics and linguistics in Regensburg, Germany. Dr. phil. Development of information systems in business and administration.

Since 1990 professor for information systems at the Department of Computer Science and Information Systems, Georg Simon Ohm University of Applied Sciences of Nuremberg, Germany.

2005-2008 visiting professor for information systems at the School of Mathematics and Systems Engineering, University of Växjö, Sweden.

Research fields: information systems modeling based upon epistemology (evolutionary epistemology, critical realism, moderate constructivism); inflectional morphology and data mining.

Felix Paetzold

Born 1976. Studies of computer science at Georg Simon Ohm University of Applied Sciences of Nuremberg, Germany. Master's degree in computer science.

Special subject: SAP.

Department of Information and Communication, City of Nuremberg, Nuremberg, Germany

Robert Breun

Born 1979. Studies of computer science at Georg Simon Ohm University of Applied Sciences of Nuremberg, Germany. Master's degree in computer science.

Special subject: IT organization.

IT Department, Bruder Spielwaren GmbH + Co. KG, Fürth, Germany