2011

Analysis of Two Theoretical Perspectives on Information Systems Development: Towards an Integrated Perspective

Frank Zickert
Goethe University Frankfurt, mail@frankzickert.de

Analysis of Two Theoretical Perspectives on Information Systems Development: Towards an Integrated Perspective

Frank Zickert
Chair of Business Administration, esp. E-Finance and Services Science
Goethe University Frankfurt, Grüneburgplatz 1,
60323 Frankfurt, Germany
+49 179 4885644
mail@frankzickert.de

ABSTRACT
In this paper, we analyze two theoretical perspectives and investigate their explanatory power on information systems development (ISD) projects. Building upon a case study, we illustrate that the perspectives of ISD as an economic transformation process and ISD as complex problem solving address different but complementary ISD phenomena. By integrating both theoretical perspectives, we are able to analyze and predict more ISD phenomena than each of the theories individually. Therefore, the contribution of this paper is twofold. Firstly, it supports researchers in their selection of a theory when addressing ISD phenomena. Secondly, it serves as an example of how researchers can develop a new theoretical perspective to address a phenomenon of interest not covered appropriately by existing theories.

Categories and Subject Descriptors
D.2 [Software Engineering]: Requirements/Specifications.

General Terms
Design, Theory.

Keywords
Theoretical Perspectives, Information Systems Development, Economic Transformation Process, Complex Problem Solving.

1. INTRODUCTION
Research in information systems development (ISD) provides us with numerous theories that explain how ISD works. These theories frame our understanding of phenomena in and around ISD. Two of the most commonly used theories are, for example, ISD as an economic transformation process [33], in which resources are used to transform the requirements of a system into a working code [20], and ISD as complex problem solving [9], in which the solution is sought by generating and evaluating alternatives of the system under construction [41].

Although diversity in theory can be useful to ISD research [44], it confronts the researcher with the problem of deciding on which theory to use for the investigation of a phenomenon of interest [57]. This decision is crucial, since the phenomena that command our attention are linked inextricably to the theories and paradigms we use to understand the world [34]. Consequently, an inappropriate selection of a theory may result in the inability to investigate the phenomenon of interest.

The purpose of this paper is to provide support for researchers in their selection of a theory when addressing ISD phenomena. Therefore, this paper aims at answering the research question of which ISD phenomena can be addressed appropriately by using the theories of ISD as an economic transformation process and ISD as complex problem solving. Moreover, by integrating both theories with each other, this paper aims to extend the scope of ISD phenomena beyond what can be addressed by either theory alone. The empirical basis for the evaluation of the theories is a software development project that we were able to investigate in a large financial institution.

The remainder of this paper is organized as follows. We depict the theories of ISD as an economic transformation process and ISD as complex problem solving in section two. After that, in section three, our case study illustrates the different insights that these theories disclose. Subsequently, in section four, we integrate both theories and return to our case applying the integrated theory. Subsequent to a brief discussion in section five, we conclude by noting benefits and limitations associated with this analysis in section six.

2. TWO THEORIES
In this paper, theories are regarded as lenses through which we see problems and observe phenomena [8]. Following this notion, theories provide explanations of how phenomena are related to the problem and from which predictions can be derived or the problem can be solved [25]. Theories are thus tools that researchers use in order to investigate phenomena of interest. Since they are tools, there is not one single correct theory that implies all others are wrong, but rather any theory can at best be appropriate or inappropriate for the investigation of a specific phenomenon. Table 1 depicts a collection of ISD theories.

This paper focuses on two theories, ISD as an economic transformation process and ISD as complex problem solving. Both theories belong to the functionalist paradigm, in which "the

10th International Conference on Wirtschaftsinformatik,
16th - 18th February 2011, Zurich, Switzerland

economic reality (translated into quantitative financial goals, and systems performance characteristics) allows system objectives to be derived in an objective, verifiable, and rational way [and where] systems design becomes primarily a technical process" [28]. Thus, both theories share fundamental assumptions about the nature of ISD, such as that in ISD, social order is used to find consensus on a solution that is the rational choice because it satisfies goals [14].

Table 1. Theories on ISD

Theory	Short Description
Economic Transformation Process	The system is transformed from objective goals into subsequent forms, such as requirements and code. [2][5][7]
Complex Problem Solving	The system is a set of parameters for which a configuration must be found that results in the desired system behavior. [9][40]
Knowledge-based	Systems are created using the aggregated knowledge of stakeholders. The team process needs to be coordinated. [23]
Negotiation	The system serves as means to the individual objectives of the stakeholders. The system characteristics are determined by negotiation. [10][42]
Complex Adaptive Systems	The system emerges as a result of the individual behavior of agents and their local optimization processes. [9][29]

The theories of ISD as an economic process and ISD as complex problem solving have been selected in this paper since both are widely used and acknowledged (e.g. [6][7][9] [21][37]).

2.1 Economic Transformation Process

The theory of ISD as an economic transformation process builds upon the economic theory of the firm that provides a formal description of the relationship between the quantity of outputs produced and the input resources employed. In the ISD process, input factors including labor (the programming team) and capital (tools and techniques) are transformed into outcomes such as new or modified software [7] as depicted in Figure 1.

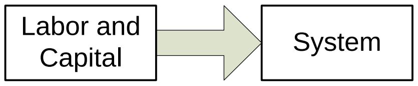

Figure 1. Software development project as a collection of transformation activities.

The central assumption underlying this perspective is a direct relationship between the input factors and the outcomes. For example, Banker et al. [7] apply the transformation process perspective in order to assess the effect of code generators or packaged software on productivity of the ISD maintenance process. Although not explicitly mentioned, Agrawal and Chari [2] build upon the notion of ISD as an economic transformation process when investigating the effects of high process maturity on outcomes, such as effort, quality, and cycle time. Anda et al. [5] quantify the impact that variations and reproducibility in the ISD process have on the quality of software projects in terms of delivery within budget and on the quality of the product in terms of functionality, reliability, usability, efficiency, maintainability, and portability.

The main phenomenon of interest of ISD as an economic transformation process is the productivity of the ISD process and related attributes, such as effort or cycle time [6]. Insights about ISD productivity are crucial since the technical ISD process is an engineering task of creating cost effective solutions to practical problems [52]. The purpose of this perspective is thus to support the creation of cost effective solutions.

In order to measure productivity, both inputs and outputs need to be measured. The most important output is represented by the system size, which can be measured by the number of function points [3], [16], a metric of business systems functionality [4], [42] or by the number of source lines of code [58], [46], [12]. Labor, as the most important input factor, is represented by the project effort, which results from the time and number of staff that are needed to build the system [1]. Moreover, both input and output factors are homogenous.

Another important input factor that is missing in this notion are the requirements of the system under construction. Although labor is also required for the elaboration of requirements, there are conceptual differences between the requirements of the system and the labor required for building the system [23]. Requirements correspond to the system under construction [56]. Just as source code, requirements are a representation of the system. Each representation of the system serves a specific purpose, has an intended audience, and has its own language. While the purpose of source code is to run on a computer and developers write it in programming languages, the purpose of requirements is to describe what the system does in its environment [29]. Requirements are socially constructed and negotiated by stakeholders as means to satisfaction of their goals [47], [10] and requirements are written in natural language [32] or specific notation languages, such as KAOS [55] or Problem Frames [50].

In a refined notion of a transformation process, the purpose of ISD is the transformation of an early representation of the system, such as requirements, into a working instance that is represented by compiled and tested source code. Since all representations correspond to the same system, correctness of the transformation can be evaluated by a direct comparison of whether the representations are congruent [19], for instance, do the requirements that describe what the system is supposed to do match with what the source code of the system actually does when it is executed.

On the contrary, other input factors, such as labor, are not actually transformed but rather consumed by transforming one representation of the system into another [13]. Labor and other consumables are thus not added to the system, but these factors refer to the ISD project in which they are consumed.

Another class of input factors comprises tools and techniques, which are neither transformed nor consumed. Tools and techniques are used within the transformation process through which system representations are transformed by using labor. Both the amount of required consumables for a transformation and the quality of a transformed system representation depend on the employed tools and techniques [59]. For instance, using a complex technique for the formal elaboration of requirements may require more labor than an easy and informal technique does. When using formal techniques however, the quality of the resulting requirements may be improved.

Figure 2 depicts the refined notion of ISD as an economic transformation process, which distinguishes between these three classes of input factors. In fact, this is still an abstract notion of ISD. The ISD process determines which specific activities are accomplished at all, whether they are done sequentially or concurrently, which representations of the system are produced, and at which points consumables are required. The waterfall model [48] serves as a blueprint of an ISD process from the perspective of ISD as an economic transformation process.

The theoretical perspective of ISD as an economic transformation process treats ISD as a black box, which means that there is no further analysis of how the transformation specifically works. On the contrary, since the input factors are homogenous, it is assumed that the transformation is repeatable and therefore predictable. That means that the ISD can be repeated with the same productivity each time it is executed. Consequently, if the ISD productivity has already been assessed, it is possible to forecast required labor for the transformation of specific systems. Cost and effort estimation methods, for example, build upon this assumption when they estimate the labor that is required for the system development based on the system size [11].

The following Table 2 summarizes the major characteristics of the theoretical perspective of ISD as an economic transformation process.

Table 2. ISD as an Economic Transformation Process

Purpose	- Creation of cost effective solutions - Effort estimations
Treats ISD as	- Black Box
Assumptions	- Direct relationship between input factors and outcomes - Input factors are homogenous - The result of transformation is predictable - Transformations are repeatable
Input factors	- Resources/Labor - Process model - System content
Phenomenon of interest	- Productivity of the ISD process

2.2 Complex Problem Solving

Another theoretical perspective on ISD is ISD as complex problem solving [9]. This perspective mainly aims at disclosing what needs to be done in order to find a satisfactory solution for the problem [41].

Marengoa and Dosi [37], for example, find in their investigation of the degree of decentralization in problem solving that decentralized structures are unlikely to generate optimal solutions if the problem is complex. Duimering et al. [21] examine the influence of product requirement ambiguity on the task structures of the development project. Their results highlight the role of communication, coordination, and knowledge as distributed development project teams struggle to resolve ambiguity. Espinosa et al. [22] investigate the effect of familiarity on how long the development team requires in order to find an error free solution to the problem.

The theoretical perspective of ISD as complex problem solving builds upon the notion of a parametric representation, in which the system is regarded as a collection of parameters. The behavior of the system, once it is completed, depends on the set of values that are assigned to the parameters. The objective in ISD is to define values for all parameters of a system in a way that results in the desired behavior of the system [31]. The complexity of finding appropriate values for all parameters originates from interrelations among the parameters [51]. Due to interrelations, whether a specific value for a parameter is valid depends on the value itself and also on values that have been assigned to related parameters.

In ISD, the problem to be solved is represented by requirements that describe what the system has to accomplish [29]. The problem is solved if all requirements are met. Requirements engineering (RE) methods, such as KAOS, support the elaboration and verification of requirements in a way that assures the requirements appropriately address the superordinate problem [18]. Therefore, requirements can be regarded as the parameters of the problem.

Subsequently, in ISD, specifications that describe how the system works are designed in order to accomplish the requirements. The specifications therefore serve as values for the parameters. Other RE methods, such as problem frames, support the correct derivation of specifications from requirements and therefore aim at assuring that only valid values are assigned to the parameters [50].

Solving a problem requires assigning valid values to all parameters. The assignment is not carried out randomly but follows a search procedure that aims at favorable configurations for the values of the parameters. These search procedures are called heuristics [41] and are well covered by literature on artificial intelligence [36], [49]. Heuristics usually converge towards a solution, which means that they do not instantly find the right configuration but start with a configuration and alter it in a way that approaches the final solution. For example, the hill climbing heuristic starts with a random configuration of parameter values and then iteratively changes parameter values. Changes that improve the resulting solution performance are kept, whereas changes that decrease resulting performance are withdrawn. As depicted in Figure 3, the performance is increased

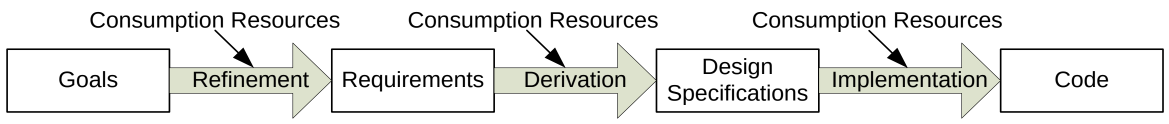

Figure 2. Software development project as a collection of transformation activities.

until an optimum is found, from which each parameter change results in a lower performance. However, depending on the starting point, the hill climbing heuristic may become stuck in local optima that may not achieve the desired performance output. In such cases, in order to find a satisfactory solution, the current path must be left and a completely different must be taken. This is done by backtracking, in which new values are assigned to parameters that have already been set in another way.

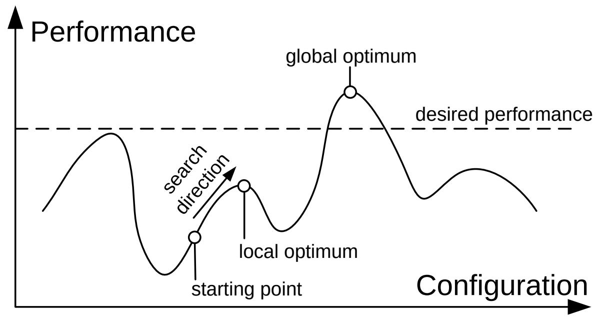

Figure 3. Hill climbing heuristic.

An underlying assumption of the perspective of ISD as complex problem solving is the decomposability of the problem. In order to be able to search for parameter values that solve the problem, the problem first has to be decomposed into a set of parameters. Based on Simon [51], problems often exhibit ‘near decomposability’, which refers to the idea that there are groups of problem components that have a high degree of interdependence to each other, whereas they are only loosely coupled with other groups of components. These groups appropriately serve as parameters, since they are relatively independent and thus it is easier to find valid values for them [21].

However, since the parameters remain interrelated with each other to a certain extent, the performance of a configuration results from the combination of parameter values, where even small changes in one parameter value can result in significant changes in the overall performance [54], [38]. As a result, in order to achieve satisfactory performance, assigning a value to a parameter may also require other parameters to take specific values. However, if the other parameters already have values that do not correspond to the required ones, some already set values must be changed respectively and reassigned. This reassignment—or backtracking—can also require other parameter values to change. Thus, it may result in cascade effects that require the complete configuration to change [10].

An assumption underlying the theoretical perspective of ISD as complex problem solving is the specificity of problems, where each problem is decomposed into a specific set of parameters [51]. Since each set of parameters exhibits a specific structure with regard to how the parameters are interrelated with each other, there is no general best way of how to solve a problem, but the performance of the applied heuristic depends on its fit to the problem structure [40]. For example, due to its property of getting stuck in local optima, hill climbing is an inappropriate heuristic for solving a problem with many local optima that do not achieve satisfactory performance. Other heuristics that do not get stuck in local optima, such as genetic algorithms, would find better solutions for such problems.

The underlying assumption is that activities are not generally repeatable but that it depends on the specific parameters whether valid values exist. In order to account for the specificity of problems and how they are solved, this perspective treats ISD as a white box. Moreover, since specific values are assigned to specific parameters, how such assignments affect the solution performance is not predictable unless they are given a try.

Table 3 summarizes the major characteristics of the theoretical perspective of ISD as complex problem solving.

Table 3. ISD as Complex Problem Solving

Purpose	- Find a satisfactory solution
Treats ISD as	- White Box
Assumptions	- Each problem is specific - Decomposability of the problem - The effect of an activity on solution performance is unknown unless it is tried - Direct manipulation of parameters and values possible
Input factors	- Problem - Heuristic/the way of how the problem is solved
Phenomenon of interest	- Performance of the system under construction

3. ISD CASE

3.1 Case Study Design

In order to get first hand information about the phenomena that the theoretical perspectives on ISD investigate, we applied a case study on a software development project in a large financial institution.

The observed project involved various stakeholders and affected different systems. Moreover, the project comprised reengineering of an existing system and its integration with another recently built system. The project was selected since the variety of both participating stakeholders and involved systems promised to be fruitful for making a distinction between two different theoretical perspectives in use.

The most important source of data was observations that we made by accompanying the business and technical analysts when requirements and design specifications were elaborated. We spent 103 hours over 40 days with the analysts on the project. During this period, we frequently had discussions with the analysts. Moreover, we were able to also interrogate other stakeholders in the project, such as the retail customer division whose representative acted as internal customer, the project manager, developers, representatives of the vendors, and members of the testing team. Moreover, sources of data also included access to documents, including working versions und reviews to the documents, such as concepts, meeting minutes, and e-mails. In total, this documentation comprised 323 pages.

Following Yin [60], we matched our data to the elements and characteristics of both theories in order to identify patterns in the data that disclose whether the project took place in the way the theories suggest. Based on this, we analyze how the theoretical perspectives explain the observations and which kind of insights each theoretical perspective supports.

As proposed by Miles and Huberman [39] we conducted data gathering and analysis concurrently so that we were able to capture all information that we found necessary for matching patterns to the theories.

3.2 Description of the Case

The project started in January 2009 and was completed in August 2009. In the project, a front-end system had to be integrated with a recently built payment processing system and therefore required reengineering. Previously, payment orders once entered at the front-end were transferred to a legacy processing system. Since the legacy processing system was planned to be deactivated, orders needed to be transferred to the new processing system instead. Moreover, since the new processing system required different data and a different payment order format than the old processing system, the order entry at the front-end had to be changed completely, wherefore it was decided to reengineer the whole system.

The applied ISD process generally adhered to the waterfall model [48]. At the beginning of the project, the business analyst collected the objectives of the retail customer division representative, which served as a basis for the elaboration and formulation of the requirements that had to be met to satisfy the objectives. All requirements were collected in a requirements document.

Since the retail customer division representative had many issues concerning the functionality of the front-end, the analyst elaborated various requirements. In order to achieve these requirements, significant changes in the front-end design were made. For example, it was requested that payment orders had to be already checked for correctness at the front-end. Since this involved verification, whether entered bank codes are valid, access to a complete list of all allowed bank codes was required.

The business analyst evaluated different alternatives for the requirement of integrating the front-end with the processing system. Using reasoning of lower maintenance effort for the future front-end system, he selected a direct interface between the front-end system and the processing system

Based on the requirements document, two technical analysts derived the software design specifications and prepared the software design specification document. While the business analyst is part of an IT department that is aligned with the retail customer division, the technical analysts are assigned to specific IT systems. Thus, in this project, there was a separate technical analyst involved for each affected system, one for the front-end system and one for the processing system.

Reengineering the front-end system not only aimed at providing the new functionality but also was intended to straighten its design. Since the front-end system already had run for several years and had undergone frequent changes, its design was quite tangled. Consequently, in order to obtain a system that is not bound to legacy structures, reengineering the front-end system started from scratch [26], only adhering to the given requirements but ignoring any constraints given by the existing systems. This however, also included leaving unconsidered the constraints given by the processing system.

Starting from scratch, the technical analyst responsible for the front-end system addressed the requirement of a direct connection between the front-end and the processing system by specifying a web-service interface. The integration of the front-end and the processing system proceeded after the front-end design was completed. Subsequent to reviewing the specified front-end design, the technical analyst, who was responsible for the processing system, rejected the implementation of a web-service interface at the processing system side, since it would not be implementable within the given constraints in time and budget. Instead, he suggested a file transfer. However, the architecture of the front-end system did not support file transfers in the suggested way. The inability to provide a web-service on the one hand and the inability to transfer requested files on the other hand not only required rework of the already specified front-end system design, but it also rendered unworkable the requirement of the direct connection between both systems. Therefore, the requirement of an indirect connection replaced the direct connection requirement, although it implied higher maintenance costs.

Subsequent to the resolution of this issue and rework of the front-end design, the on-site developers and the external vendor developed the software code based on the software requirements specifications document. Subsequently, the testing team performed the software tests. Despite some minor bug-fixings, neither code development nor testing disclosed any problems that required considerable rework.

3.3 From an Economic Transformation Process Perspective

The most significant observation regarding the project at hand from the theoretical perspective of ISD as an economic transformation process is provided by the organizational policy that specifies all activities and their outcomes in the project. According to this policy, each activity has to have a described output that is the input for the next activity. For example, in the requirements analysis phase, requirements had to be elaborated by the business analyst and had to be written down in natural language. A template for the requirements document had to be used, which provides a document structure and the required contents. This requirements document served as input for the design specification phase, in which the technical analysts derived design specifications from the requirements. A software design specifications document had to be produced, whose content was also pre-structured by a template that had to be used. The design specifications were handed over to the developers and the external vendor, who prepared the source code, which was finally handed over to the testing team.

Each of these produced outputs referred to the system under construction and is thus a representation of this system. The process model described which activities the employees had to accomplish and which tools (e. g. templates) they had to use. The required labor for accomplishing the activities was gathered using the organizational accounting tool, which every employee used to charge the spent working time to a project.

Altogether, these factors do not only support the assessment of the ISD productivity in this project, they also allow identification of productivity drivers. For example, the required formalism in the activities consumed a significant amount of labor and thus negatively affected productivity. Although the analysts had delineated requirements and design specifications using self-made models or descriptions, they had to spend about the same amount of work filling out the required template documents. Other factors that affected the productivity were the number of requirements that needed to be transformed into the working system and the number of required attempts for the correct transformation. The business analyst, for instance, elaborated different alternative integration requirements of the connection between the front-end and the processing system. Each of these alternatives needed to be elaborated and described and thus required labor that reduced productivity. Reworking the integration requirement after the web-service had been rejected is another example of a factor that negatively affected productivity.

However, although this perspective allows the identification of factors that affect productivity, such as required rework, it does not explain why rework occurred. Building upon the assumption that both input and output of an activity are homogenous, the investigation of any specific input or output is unsupported from this theoretical perspective.

3.4 From a Complex Problem Solving Perspective

Despite general adherence to the given organizational policy, the project at hand was not accomplished in a unidirectional and straightforward manner, but could be characterized as a continuous search, in which different alternatives were evaluated in order to find a solution that exhibited the requested performance characteristics.

At the beginning of the project, for example, the business analyst considered different alternative requirements before he was able to determine that a direct interface between the front-end system and the processing system is the requirement with the best performance attributes since it resulted in low maintenance cost. In fact, however, this of all requirements turned out to be inappropriate for a satisfactory solution, because it was not accomplishable. The designed web-service could not be integrated with the processing system within the given constraints in time and budget and the file transfer that would have worked with the processing system did not work with the front-end system.

In this situation, the integration requirement of a direct connection served as a parameter that comprised a dependency between the interfaces at the front-end and the processing system. Because of this dependency, a design specification that represents a value of this parameter had to work with both systems. However, although assigning a value that worked with both systems to this parameter was impossible in this situation, the selection of this parameter was not per se false. In fact, two design specifications could have achieved the requirement and thus depicted valid values for this parameter. What made this requirement unworkable was the dependency, due to which both the front-end and the processing system had to share the same value. While the web-service specification did not work with the processing system, the file transfer specification did not work with the front-end system. Thus, the inappropriateness of the requirement of a direct connection was not disclosed until design specifications were derived from it.

In order to solve the problem, despite the inconsistency among the required values for the direct connection requirement, the dependency between the values had to be resolved. Backtracking the direct connection requirement and replacing it with the requirement of an indirect connection decoupled both systems from each other and therefore enabled solving the problem. Without having had the chance to withdraw the requirement of the direct connection, the problem would not have been solvable. It would have resulted in failure of the project. Although the requirement of the indirect connection created other dependencies, such as the interfaces to a routing system, these new dependencies did not result in any problems with regard to finding appropriate design specifications as values.

This perspective offers insights about which specific activities and decisions in ISD were required in order to find a satisfactory solution. It provides an explanation of why specific requirements and design specifications had to be reworked in our case. For example, it discloses that the inconsistency between required values for the requirement of the direct connection inhibited solving the problem.

However, this perspective does not put the decisions made for solving the problem into an ISD context that explains why inconsistencies occurred at all, for example, whether the reason for the inconsistency was the ISD process, insufficient resources, or the problem of building a system.

4. TOWARDS AN INTEGRATED PERSPECTIVE

4.1 Theoretical Perspective

While both the theoretical perspectives of ISD as an economic transformation process and ISD as complex problem solving support the addressing of different phenomena of interest, both perspectives also have limitations with regard to which aspects they are able to explain. While the transformation process perspective sets input factors, such as attributes to the ISD process, into relation with the produced output and therefore discloses factors affecting productivity, it does not give underlying reasons of why the factors matter. The complex problem solving perspective, on the contrary, allows investigating the structures underlying ISD and therefore provides insights into why specific problems occur in a project. However, it does not put these problems into relation to attributes of the ISD process, and therefore, it fails to provide measures on how to improve ISD.

Since the phenomena of interest that the theoretical perspectives address are complementary, an integrated perspective that combines both theories may address phenomena of interest beyond the phenomena addressed by either theory alone. Moreover, both theories share the same fundamental assumptions about the world, because they both build upon the functionalist paradigm.

The integrated perspective regards ISD as a collection of solution space transformation activities. The solution space contains all potential solutions to the problem, regardless of whether their performances are satisfactory or not [10]. Like the complex problem solving perspective, the integrated perspective builds upon the notion of a parametric representation of the system under construction, in which the configuration of the parameter values results in the behavior of the system once it is built. Therefore, the solution space contains all configurations of parameter values.

However, in contrast to the complex problem solving perspective, in which ISD takes place as a conscious search for the parameters and their values, in the integrated perspective, accomplished activities unconsciously determine the parameters and their values, as is explained in the following.

Seen from the perspective of ISD as complex problem solving, parameters and their values are directly manipulated and therefore the configurations whose performance is sought to be evaluated are known. Although the performance of a configuration is unknown unless it is evaluated, heuristics calculate configurations worth consideration based on the performance of already evaluated configurations. For example, the genetic algorithm heuristic generates promising configurations by recombining parts of configurations with good performance [36].

On the contrary, seen from the integrated perspective of ISD as a collection of solution space transformation activities, only activities are consciously selected, whereas the configuration of parameters and their values results from the activities in an unpredictable way. That means, not only the performance of a configuration but also the specific configuration is unknown unless the activity that results in the respective configuration is accomplished. As a result, it is impossible to employ a heuristic because it is impossible to generate specific configurations selectively. Therefore, it is not a heuristic but the current situation in the ISD project that supports decisions on which activities to execute and which resources to employ in order to solve the given problem.

In this regard, the integrated perspective is similar to the perspective of ISD as an economic transformation process. There are specific activities in ISD that are executed in order to build the system and each activity requires resources—most importantly labor. However, while in the notion of the economic transformation perspective, activities directly transform the content of the final solution in a predictable way, in the notion of the integrated perspective, activities transform the current configuration in an unpredictable way.

Since the current configuration determines which other configurations can be achieved by performing further activities, one needs to distinguish between the actual solution space that only contains solutions that are achievable from the current configuration and the overall solution space that contains all configurations.

The actual solution space evolves over time. With each activity, it approaches a solution which, however, is unknown both in terms of its configuration and its performance. Therefore, whether the solution exhibits satisfactory performance is unknown, too. Since neither the configuration nor its performance are predictable, although both depend on the activities, the employed resources, and the specific problem, ISD is not directed in any way, neither in terms of conscious problem solving, nor in terms of simply transforming the content of the system under construction. The phenomenon of interest of the integrated perspective therefore is to investigate why ISD is successful or fails at all.

Table 4. ISD as a Collection of Solution Space Transformation Activities

Purpose	- Investigations of the structures underlying ISD and putting them into relation with general input factors
Treats ISD as	- White Box
Assumptions	- Each problem is specific - Decomposability of the problem - Activities are repeatable, but their outcome is not predictable because it depends on the content and the employed resources
Input factors	- Resources/Labor - Composition of activities - System content
Phenomenon of interest	- Reasons for ISD success or failure

By investigating how the actual solution space evolves in a project, this perspective allows tracing back problems, such as inconsistencies, to their origins. This perspective discloses whether the origin of success or failure in a specific case is the process model, the employed resources, or an unsolvable problem. Table 4 summarizes the major characteristics of the theoretical perspective of ISD as a collection of solution space transformation activities.

4.2 THE CASE REVISITED

The most significant characteristic of the observed case, which supports the notion of ISD as a collection of solution space transformation activities, is that the solution space was unknown. At no time, did decision makers consciously take into account how many or which solutions the actual solution space comprised. However, we will particularly consider the actual solution space in the following, when applying the integrated perspective on the case.

The major problem in the project at hand became evident when the front-end system was integrated with the processing system. In this situation, the actual solution space contained no valid solution. Although there were two considered solutions, the web-service interface as suggested by the technical analyst who was responsible for the front-end system design and the file transfer suggested by the technical analyst responsible for the processing system design, no solution worked with both systems. Therefore, both solutions were invalid, leaving no valid solution in the actual solution space.

In order to look into the cause for this “emptiness” of the actual solution space that resulted in backtracking and thus rework, related activities are analyzed. The design of the web-service was the proximate activity, in which the technical analyst's task comprised addressing the requirement of a direct connection when reengineering the front-end system from scratch. The analyst successfully accomplished the task since the designed web-service appropriately addressed the direct connection requirement. It represented a valid solution for the given sub-problem.

Since the analyst successfully accomplished the design task, the reason that caused the empty actual solution space is not the analyst's fault but rather the activity itself. Particularly, the conscious neglect of the dependency to the interface of the existing processing system is questionable, because it delayed discovery of the empty actual solution space until the integration of both systems. However, since the front-end system turned out not to be able to support the file transfer as required from the processing system, even an early consideration of the dependency would not have resulted in anything but an empty actual solution space. Since neither insufficient nor incapable resources nor the neglect of the dependency caused the empty actual solution space, it must have been already empty prior to the derivation of design specifications.

Nevertheless, the division of labor affected the amount of accrued rework. Early consideration of the interdependency between the front-end and processing system interfaces would have disclosed earlier that the actual solution space was empty. It would have been recognized before a significant amount of work was spent on the complete front-end design specification. Thus, although the activity setting in this situation did not cause rework, it determined its extent. The recommendation therefore is to take into account all dependencies early.

In order to further investigate the cause of rework in this project, the activity, in which the parameters were set, needs to be analyzed. The business analyst set the parameters when elaborating the requirements at the beginning of the project. This activity aimed at requirements that can be met and, if met, satisfy the stakeholders' objectives. Although the first elaborated requirement of a direct connection could not be met, the activity was generally accomplishable as the second elaboration of the indirect connection requirement discloses. Therefore, the actual solution space at this time contained at least one valid and satisfactory solution that, however, was not selected right away.

Nevertheless, the selected requirement of a direct connection was a rational decision, because it was the best choice reflecting the information available to the business analyst at the time [17]. Firstly, the requirement of a direct connection best satisfied the objectives, because it also resulted in lower maintenance cost than the indirect connection requirement. And secondly, the information about the requirement of a direct connection to be unworkable did not emerge until the design was specified. When the business analyst first elaborated the direct connection requirement, the resulting actual solution space contained two seemingly valid solutions: the web-service and the file transfer. The decision would have been irrational only if some feasible arrangement for recognizing and achieving a preferred outcome existed, but that outcome was not obtained [35].

Since the problem of selecting an appropriate requirement was solvable and the decisions were rational in the given context, the activity that set the context for the decision needs to be critically analyzed. The given process model arranged for the final elaboration of requirements before their viability was checked further. As a result, information required in order to not only make rational but also beneficial decisions was unavailable when decisions had to be made. Therefore, the insights of this theoretical perspective recommend an ISD process that assures all relevant information be available when decisions need to be made. Concurrent requirement elaboration and design specification would make available information about requirement viability early and therefore could improve the quality of decision making [45]. However, since design specifications are built upon not yet finalized requirements, rework would occur if requirements turned out to not completely address the stakeholder objectives [53].

Altogether, the integrated perspective of ISD as a collection of solution space transformation activities suggests that good decisions are not necessarily those that best satisfy goals, but those that also allow further problem solving. In two situations within the observed project, the decisions that aimed at achieving the best solution resulted in severe consequences. Firstly, although the selected direct connection requirement would have implied lower maintenance cost, it resulted in an empty actual solution space and therefore in an unworkable situation that caused rework of the requirement and all design specifications building upon it. Secondly, although the chosen web-service interface would have implied a straightened design, it resulted in a large extent of rework.

However, this does not imply that goal satisfaction should not be a major factor for decision-making. It rather implies that the effect that decisions have on the actual solution space also needs to be included in the decision-making. For example, the requirement of the direct connection had the disadvantageous effect of coupling the interfaces between front-end and processing system and therefore increased complexity of accomplishing the activity [15]. However, much work is needed in order to assess the effect that decisions have on the actual solution space.

5. DISCUSSION

Theories allow knowledge to be accumulated in a systematic manner and this accumulated body of knowledge enlightens professional practice [25]. Therefore, the primary interest of scientific research is to add to the body of knowledge by the creation, refinement, and validity assessment of theories. However, since theories in the body of knowledge also serve as utilities from and through which IS research is accomplished [27], the researcher must be aware of the nature of the applied theories. Theories are only valid in a context that is determined by basic assumptions about the world and specific assumptions about the phenomenon of interest [28]. These assumptions must be considered when applying theories. Otherwise, findings may be misinterpreted or even void. Therefore, a critical eye on theories in the body of knowledge is required in order to not rely on serendipity when selecting a theory. Researchers need to be aware of the assumptions and beliefs that they employ in their day-to-day activities [28]. Therefore, further analyses are required in order to structure the body of knowledge in a way that makes it comprehensible and usable for subsequent research.

6. CONCLUDING REMARKS

In this paper, we investigated which phenomena of interest two widely used theoretical perspectives address in the area of ISD support. By building upon an analysis of the perspectives and a case study of a software development project conducted in a large financial institution, this paper has three findings.

Firstly, ISD productivity is the main phenomenon of interest of the theoretical perspective of ISD as an economic transformation process. While this perspective allows identifying factors affecting ISD productivity, such as rework, it does not explain the rationale underlying these factors, since it treats ISD as a black box. Therefore, it does not disclose measures positively influencing the factors, for example, measures reducing rework.

Secondly, the performance of the system under construction is the main phenomenon of interest of the perspective of ISD as complex problem solving. Since this perspective treats ISD as a white box, it supports investigations of how decisions in the ISD process affect performance. For example, it discloses that backtracking is vital for finding satisfactory solutions. However, this perspective does not put the decisions made for solving the problem into the specific ISD context. Therefore, it does not support conclusions on whether decisions, such as to backtrack, are reasoned with the ISD process, insufficient resources, or the problem of building a system.

Thirdly, an integrated perspective that combines both ISD as an economic transformation process and ISD as a complex problem solving, supports addressing the underlying reasons for ISD success or failure. In our case, the integrated perspective discloses that the applied process caused rework and determined its extent. Based on the insights that the integrated perspective provides, measures positively influencing ISD success can be identified, for example, making information about the consequences of decisions available as early as possible.

Our findings about which ISD problems can be addressed by using which theoretical perspectives provide support for researchers in their selection of a theoretical perspective when investigating ISD problems. Moreover, by integrating two theories, the paper serves as an example of how researchers can prepare a theoretical lens that is suited for the investigation of a phenomenon of interest that is not appropriately addressed by one single perspective.

However, this analysis has some limitations that future work needs to address. Firstly, the scope of this analysis is limited to the evaluation of two theoretical perspectives on ISD within the functionalist paradigm. There are in fact other theoretical perspectives within this or within other paradigms, which still have to be critically analyzed. Future work needs to evaluate these theories in order to create a framework that researchers can use when selecting a theory.

Secondly, although this analysis of the theoretical perspectives also provides some insights on ISD, it has to be noticed that these insights build upon a single case. In fact, we do not claim to have gathered any statistically generalized insights but rather analytical ones. In this paper, the insights on ISD illustrate which kind of insights the theoretical perspectives can provide. Nevertheless, the insights on ISD seem interesting and therefore deserve further scientific investigation.

7. Acknowledgements

This work was developed as part of a research project of the E-Finance Lab at Goethe University Frankfurt. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the E-Finance Lab or its supporting partners. The authors are indebted to the participating universities and gratefully acknowledge all support of their industry partners.

8. REFERENCES

[1] Abdel-Hamid, T. and Madnick, S. 1991. *Software Project Dynamics: An Integrated Approach*. Prentice-Hall, Englewood Cliffs, NJ.

[2] Agrawal, M. and Chari, K. 2007. Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects. *IEEE Transactions on Software Engineering* 33, 3, 145-156.

[3] Albrecht, A. 1979. Measuring Application Development Productivity. *Proceedings of IBM Applications Development Symposium*, 83-92.

[4] Albrecht, A. and Gaffney, J. 1983. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering* 9, 6, 639-648.

[5] Anda, B., Sjøberg, D., and Mockus, A. 2009. Variability and Reproducibility in Software Engineering: A Study of Four Companies that Developed the Same System. *IEEE Transactions on Software Engineering* 35, 3, 407-429.

[6] Banker, R., Datar, S., and Kemerer, C. 1991. A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects. *Management Science* 37, 1, 1-18.

[7] Banker, R., Davis, G., and Slaughter, S. 1998. Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study. *Management Science* 44, 4, 433-450.

[8] Benbasat, I. and Weber, R. 1996. Research Commentary: Rethinking ‘Diversity’ in Information Systems Research. *Information Systems Research* 7, 4, 389-399.

[9] Benbya, H. and McKelvey, B. 2006. Toward a Complexity Theory of Information Systems Development. *Information Technology & People* 19, 1, 12-34.

[10] Bergman, M., King, J., and Lyytinen, K. 2002. Large-Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering. *Requirements Engineering* 7, 3, 152-171.

[11] Boehm, B., Abts, C., and Chulani, S. 2000. Software Development Cost Estimation Approaches – A Survey. *Annals of Software Engineering* 10, 177-205.

[12] Boehm, B. 1981. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ.

[13] Boehm, B. 2007. Improving software productivity. In: *Software Engineering*, R. Selby, Ed. John Wiley and Sons, Hoboken, NJ, 151-178.

[14] Burrell, G. and Morgan, G. 1979. *Sociological Paradigms and Organizational Analysis*. Heinemann, London.

[15] Campbell, D. 1988. Task Complexity: A Review and Analysis. *The Academy of Management Review* 13, 1, 40-52.

[16] Cheung, Y., Willis, R., and Milne, B. 1999. Software Benchmarks Using Function Point Analysis. *Benchmarking: An International Journal* 6, 269-279.

[17] Cyert, R. and March, J. 1963. *A Behavioral Theory of the Firm*. Blackwell, Cambridge.

[18] Dardenne, A., Fickas, S., and van Lamsweerde, A. 1991. Goal-directed concept acquisition in requirements elicitation. *Proceedings of the 6th international workshop on Software specification and design*, Como, Italy, 14-21.

[19] Darimont, R. and van Lamsweerde, A. 1996. Formal refinement patterns for goal-driven requirements elaboration. *Fourth ACM SIGSOFT Symposium on the Foundations of Software Engineering*, San Francisco, 179-190.

[20] Davis, A. 1993. *Software Requirements: Objects, Functions, and States*. Prenctice-Hall, Englewood Cliffs, NJ.

[21] Duimering, P., Ran, B., Derbentseva, N., and Poile, C. 2006. The Effects of Ambiguity on Project Task Structure in New Product Development. *Knowledge and Process Management* 13, 4, 239-251.

[22] Espinosa, J., Slaughter, S., Kraut, R., and Herbsleb, J. 2007. Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development. *Organization Science* 18, 4, 613-630.

[23] Faraj, S. and Sproull, L. 2000. Coordinating Expertise in Software Development Teams. *Management Science* 46, 12, 1554-1568.

[24] Glinz, M. 2007. On Non-functional Requirements. *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'07)*, 21-26.

[25] Gregor, S. 2006. The nature of theory in IS. *MIS Quarterly* 30, 611-642.

[26] Hammer, M. and Champy, J. 1993. *Reengineering the Corporation*. Harper Collins, New York, NY.

[27] Hevner, A., March, S., Park, J., and Ram, S. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28, 75-105.

[28] Hirschheim, R. and Klein, H. 1989. Four Paradigms of Information Systems Development. *Communications of the ACM* 32, 10, 1199-1216.

[29] Holland, J. 1996. *Hidden Order, How Adaption Builds Complexity*. Basic Books, New York, NY.

[30] Jackson, M. 1995. *Software Rrequirements and Specifications*. Adison-Wesley, New York, NY.

[31] Kauffman, S. 1995. *At Home in the Universe*. Oxford University Press, New York, NY.

[32] Kotonya, G. and Sommerville, I. 1998. *Requirements Engineering: Processes and Techniques*. John Wiley and Sons, New York , NY.

[33] Kriebel, C. and Raviv, A. 1980. An Economics Approach to Modeling the Productivity of Computer Systems. *Management Science* 26, 3, 297-311.

[34] Kuhn, T. 1970. *The Structure of Scientific Revolutions*. University of Chicago Press, Chicago, IL.

[35] Liebowitz, S. and Margolis, S. 1995. Path Dependence, Lock-in, and History. *Journal of Law, Economics, and Organization* 11, 1, 205-226.

[36] Luger, G. 2002. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley, Reading, MA.

[37] Marengoa, L. and Dosi, G. 2005. Division of Labor, Organizational Coordination and Market Mechanisms in Collective Problem-solving. *Journal of Economic Behavior & Organization* 58, 303-326.

[38] McCann, J. and Ferry, D. 1979. An Approach for Assessing and Managing Inter-Unit Interdependence. *The Academy of Management Review* 4, 1, 113–119.

[39] Miles, M. and Huberman, 1994. *A Qualitative Data Analysis*. Sage Publications, Thousand Oaks, CA.

[40] Newell, A. and Simon, H. 1972. *Human Problem Solving.* Prentice Hall, Englewood Cliffs, NJ.

[41] Newell, A. and Simon, H. 1976. Computer Science as Empirical Enquiry: Symbols and Search. *Communications of the ACM* 19, 113–126.

[42] Ovaska, P, Rossi, M., and Smolander, K. 2005. Filtering, Negotiating and Shifting in the Understanding of Information System Requirements. *Scandinavian Journal of Information Systems* 17, 1, 31-66.

[43] Perry, W. 1986. The Best Data Processing Measures. *System Development* 6,6, 4-6.

[44] Pfeffer, J. 1993. Barriers to the Advance of Organizational Science: Paradigm Development as a Dependent Variable. *Academy of Management Review* 18, 4, 599-620.

[45] Pich, M., Loch, C., and de Meyer, A. 2002. On Uncertainty, Ambiguity, and Complexity in Project Management. *Management Science* 48, 8, 1008-1023.

[46] Putnam, L. 1978. General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Transactions on Software Engineering* 4, 345-361.

[47] Robinson, W. 1990. Negotiation Behavior During Requirements Specification. *Proceedings of the 12th international conference on Software engineering*, Nice, 268-276.

[48] Royce, W. 1970. Managing The Development of Large Software Systems. *Proceedings of IEEE WESCON* 26, 1-9.

[49] Russel, S. and Norvig, P. 2003. *Artificial Intelligence, A Modern Approach*. Prentice Hall, Upper Saddle River, NJ.

[50] Seater, R., Jackson, D., and Gheyi, R. 2007. Requirement Progression in Problem Frames: Deriving Specifications from Requirements. *Requirements Engineering* 12, 2, 77-102.

[51] Simon, H. 1996. *The Sciences of the Artificial*. MIT Press, Cambridge, MA.

[52] Shaw, M. 1990. Prospect for an Engineering Discipline of Software. *IEEE Software* 7, 15-24.

[53] Terwiesch, C. and Loch, C. 1999. Measuring the Effectiveness of Overlapping Development Activities. *Management Science* 45, 4, 455-465.

[54] Thompson, J. 1967. *Organizations in Action*. McGraw-Hill, New York, NY.

[55] van Lamsweerde, A., Darimont, R., and Massonet, P. 1995. Goal-directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt. *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering*, 194-203.

[56] van Lamsweerde, A. 2001. Goal-oriented Requirements Engineering: A Guided Tour. *Invited Paper for RE'01 - 5th IEEE International Symposium on Requirements Engineering*, Toronto, 249-263.

[57] Walsham, G. 2006. Doing Interpretive Research. *European Journal of Information Systems* 15, 320-330.

[58] Waltson, C. and Felix, C. 1977. A Method of Programming Measurement and Estimation. *IBM Systems Journal* 16, 1, 54-73.

[59] Wirth, N. 1995. A Plea for Lean Software. *IEEE Computer* 28, 2, 64-68.

[60] Yin, R. 1994. *Case Study Research: Design and Methods*. Sage Publications, Thousand Oaks, CA.