# THE ANATOMY OF A DESIGN THEORY

Shirley Gregor
College of Business and Economics
The Australian National University
Canberra ACT 0200
Australia
Phone: +61 2 61253749
Fax: +61 61254310
Email: Shirley.Gregor@anu.edu.au

David Jones
Division of Teaching and Learning Services
Central Queensland University
Rockhampton QLD 4702
Australia
Phone: +61 7 49309856
Fax: +61 7 49309729
Email: d.jones@cqu.edu.au

22 March, 2007

## *ABSTRACT*

Design work and design knowledge in Information Systems (IS) is important for both research and practice. Yet there has been comparatively little critical attention paid to the problem of **specifying design theory** so that it can be communicated, justified and developed cumulatively. In this essay we focus on the structural components or anatomy of design theories in IS as a special class of theory. In doing so we aim to extend the work of Walls, Widemeyer and El Sawy (1992) on the specification of information systems design theories (ISDT), drawing on other streams of thought on design research and theory to provide a basis for a more

systematic and useable formulation of these theories. Eight separate components of design theories are identified: (1) purpose and scope; (2) constructs; (3) principles of form and function; (4) artifact mutability; (5) testable propositions; (6) justificatory knowledge (kernel theories); (7) principles of implementation; and (8) an expository instantiation. This specification includes components missing in the Walls et al. adaptation of Dubin (1978) and Simon (1969) and also addresses explicitly problems associated with the role of instantiations and the specification of design theories for methodologies and interventions as well as for products and applications. The essay is significant as the unambiguous establishment of design knowledge as theory gives a sounder base for arguments for the rigor and legitimacy of IS as an applied discipline and for its continuing progress. A craft can proceed with the copying of one example of a design artifact by one artisan after another. A discipline cannot.

**Keywords:** design theory, design science, constructive research, philosophy of science, information systems, information technology, artifacts, theory structure

## I. INTRODUCTION

It is difficult to over-emphasize the significance of design work and design knowledge in Information Systems (IS) for both research and practice. Theories for design and action continue to be highly influential in IS, despite the fact that they are not always recognized as theories. Some seminal examples include structured systems analysis (Gane and Sarson, 1979) and the Systems Development Life Cycle (SDLC) model. Design theories also give prescriptions for the architecture of specific applications, such as decision support systems (Turban and Aronson, 2001), a type of knowledge that forms a large part of curricula in IS, software

engineering and computer science education. Moreover, this knowledge has vital relevance to practitioners working with information systems. As van Aken (2004, p. 220) argues eloquently, one needs prescription-driven research that provides solutions for management problems in addition to description-driven research that enables us to understand the nature of problems but leaves undone the task of developing sound change programs. Increasing attention is being paid to this type of research in IS, notably by March and Smith (1995) and Hevner et al. (2004). The ISWorld website now has a section on design research with a current overview provided by Vaishnavi and Kuechler (2004/5). Some major issues, however, remain relatively unexplored.

One important issue is how design knowledge is captured, written down and communicated. Herbert Simon in his seminal work *The Sciences of the Artificial* (1996, p. 113) argued that we need a science of design that is "tough, analytic, partly formalizable, partly empirical, teachable doctrine". Making design science formalizable, at least in part, means that we need to pay attention to how design knowledge is expressed as theory. Gregor (2006) shows how design theory can be seen as the fifth in five classes of theory that are relevant to IS: (1) theory for analysing; (2) theory for explaining, (3) theory for predicting; (4) theory for explaining and predicting; and (5) theory for design and action. The distinguishing attribute of theories for design and action is that they focus on "how to do something". They give explicit prescriptions on how to design and develop an artifact, whether it is a technological product or a managerial intervention. Of course, for this type of theory, as Hevner et al. (2004) show, we also need to consider epistemological questions of how knowledge is acquired and tested. This

current essay, however, does not concern research methods or research approaches, important as they are, but the ontological components of the theory itself. We are taking a meta-theoretical view of the nature of design theories in IS in general. The aim of the paper is to show the structural components (the anatomy) that are needed to communicate a design theory.

The focus of the paper is on the anatomy of design theories in the discipline of IS, although much of the underlying literature in our discussion comes from a range of disciplines and it is possible that our arguments have wider applicability. However, a characteristic that distinguishes IS from other fields is that it concerns the use of artifacts in human-machine systems. Lee (2001, p iii) uses these words:

> *Research in the information systems field examines more than just the technological system, or just the social system, or even the two side by side; in addition, it investigates the phenomena that emerge when the two interact.*

Thus, we have a discipline that is at the intersection of knowledge of the properties of physical objects (machines) and knowledge of human behavior, and it is possible that IS design theories may take on a different form from those in other disciplines. The IS discipline is increasingly seen as one concerned with the design, construction and use of artifacts based on information technology (IT), although the exact range and nature of the artifacts of interest is a matter of some debate (see Dahlbom, 1996; Orlikowski and Iacono, 2001; Benbasat and Zmud, 2003). The term *artifact* is used to describe something that is artificial, or constructed by humans, as opposed to something that occurs naturally (Simon 1996).

A central issue that must be acknowledged is that some researchers would argue with the use of the word “theory” for design-type knowledge, preferring to restrict the word to the possibly more familiar natural-science (and, later, social-science) types of theory. Gregor (2006) highlights the differences in views on what constitutes theory, and shows that there are both proponents and opponents for the five types of theory she identifies. With respect to theory for design and action, Simon (1996) is the well-recognized proponent of this form of theory and others have followed his lead (Iivari, 1983; Markus et al., 2002; Walls et al., 1992). Van Aken (2005) uses the term “Management Theory” for prescriptive, solution-oriented knowledge that encompasses “technological rules”, while distinguishing more description-oriented knowledge as “Organization Theory”.

Otherwise there is some feeling against recognizing design principles as theory. March and Smith (1995) and Hevner at al. (2004) promote design science as a research activity, but tend to reserve the word “theory” for natural-science-type research (Type 3 and 4 theory in Gregor, 2006). The seemingly different views may in part be semantic and depend on individual views of what is meant by theory, as outlined above. A broad view of theory is adopted here, congruent with Gregor (2006) and the OED (2004), which means that the term theory encompasses what might be termed elsewhere conjectures, models, frameworks, or body of knowledge - terms that are used in connection with design science by many authors. For example, Hevner et al. (2004), see “constructs, models and methods” as three of the four outputs of design science, with the “artifact” being the fourth. A broader view of theory means that the first three outputs are regarded as components of theory.

We believe that it is of vital importance to investigate how design knowledge can be expressed as theory (see also Purao, 2002; Rossi and Sein, 2003; Vaishnavi and Kuechler, 2004/5), although some might argue that the benefits of design research can be enjoyed without the need for theories of design. The weakness of this latter view is demonstrated by Cross (2001, p. 4), who deals comprehensively with the idea of design as a discipline. Cross shows how at one level design work can proceed without reflection on theory:

> *We must not forget that design knowledge resides in products themselves: in the forms and materials and finishes which embody design attributes. Much everyday design work entails the use of precedents or previous exemplars – not because of laziness by the designer but because the exemplars actually contain knowledge of what the product should be. This is certainly true in craft-base design: traditional crafts are based on the knowledge implicit within the object itself of how best to shape, make and use it. This is why craft-made products are usually copied very literally from one example to the next, from one generation to the next.*

However, we would prefer that IS rises above the level of a craft and agree with Cross, who says that in addition to this informal product knowledge, we need for design research:

> *the development of more formal knowledge of shape and configuration – **theoretical** studies of design morphology. (Cross, 2001, p. 5, emphasis added)*

Seeking to express IS design knowledge as theory means we give a sounder basis for arguing for the rigor and legitimacy of IS as an applied discipline, in comparison with the older, more traditional disciplines in the natural sciences, which use a

complementary, but different paradigm.[1]Our own experience has shown how both students and more experienced researchers struggle with the problem of expressing design knowledge in an acceptable form in theses and journal articles. Better understanding of the nature of design theory provides an avenue for the more systematic specification of design knowledge.

Furthermore, understanding the nature of IS design theories supports the cumulative building of knowledge, rather than the re-invention of design artifacts and methods under new labels in the waves of "fads and fancies" that tend to characterize IS/IT. As an example, the basic problem of understanding how to capture the tacit knowledge of experts remains much the same whether it is studied for expert systems or knowledge management systems, and whatever the application domain. Our design theories should be classified and compared under the most general statement of the problem being addressed that can be found (the purpose and scope of the theory), for example, "capturing tacit knowledge from experts in organizations". A claim for a better theory needs to show that the new theory provides an advance on all previous methods for solving this problem, no matter in which disciplinary sub-field they have been proposed. Again, personal experience has shown that this requirement is not well understood by many authors and this shortcoming is a common cause of journal papers being rejected for not making a sufficient theoretical contribution.

As an initial introduction to the idea of design theory structure, Table 1 shows how

[1] This issue in one also for many other applied disciplines such as accounting, education, management marketing, engineering, and other fields of information technology, marketing, engineering, and other fields of information technology.

our proposed anatomy of a design theory can be detected in Codd's articles introducing the relational database model. This anatomical skeleton, consisting of eight fundamental components, is what we derive more thoroughly in the remainder of this essay.

Table 1: Example of the skeleton of a design theory (from Codd, 1970, 1982)

| Article details | The design theory anatomy |
|---|---|
| The introduction says better database technology is needed to increase human productivity.<br>(Motivation is also provided: This need is significant because current approaches are failing.) | The **purpose and scope** of the theory are stated. |
| The relational database model has principles such as "the order of rows in the tables is arbitrary and irrelevant". | **Principles of form and function** incorporating underlying **constructs** (such as "table") are given. |
| The argument is made that the relational model allows for relatively simple adaptation and change to base tables, while user views appear unchanged. | **Artifact mutability** is addressed. |
| Statements are made such as "A relational database can perform as well as a non-relational database". | These statements are **testable propositions**. |
| It is shown how the relational model works, by reference to underlying set theory and also human cognitive processes. | **Justificatory knowledge** (kernel theory) is provided. |
| Guidelines are given on how to produce a relational database through normalization procedures. | **Principles of implementation** are given. |
| An illustration of working relational databases is provided. | An **expository instantiation** is given. |

The anatomy of design theories has received relatively little critical attention. Walls et al. (1992, p. 36) made a valuable initial attempt at this problem and we build on this work. These authors defined an information systems design theory (ISDT) as "a prescriptive theory which integrates normative and descriptive theories into design paths intended to produce more effective information systems". In 2004 Walls et al.

provided a retrospective on the fate of their ISDT formulation and they expressed some disappointment about what they saw as its limited use. They wondered if their specification was too unwieldy or cumbersome for general use, or too difficult to grasp, and concluded that their ISDT required "much more work in being complete and in making the exposition more palatable" (p. 55). We agree that it is timely to consider whether improvement in their specification model is possible.

The primary sources drawn upon by Walls et al. in their 1992 paper were Dubin's (1978) depiction of theory of the natural-science type and Simon's (1981) depiction of the sciences of the artificial. Perhaps not surprisingly, given the novelty of their endeavor and the dual aims of their article, these authors did not capture fully the range of ideas offered by Dubin and Simon, or ideas that have been presented in other important related work. Two of Dubin's mandatory theory components are missing from the specification by Walls et al. These components are the "units", the constructs that are the basic building blocks of theory, and "system states", the range of system states that the theory covers. The problem of specifying a theory for methodologies as opposed to a theory for a product was not explicitly addressed and their formulation had some unnecessary complexity in that it required kernel theories for design product and design process to be separated. Furthermore, Walls et al. (2004) themselves wondered if their depiction of design theory components was too unwieldy for use. They looked at the comparatively few articles that had explicitly referred to their formulation of ISDT, but did not consider the continuing over-arching tradition of presenting design-type work in our IS journals (see Gregor, 2006; Morrison and George, 1995; Orlikoski and Iacono, 2001), where there are alternative forms. The structures implicit in other design-type work in this substantial

history gives clues as to what might be more familiar and more useable ways of presenting design theories.

The contribution of the current paper is that it proposes solutions for the problematic areas of the Walls et al. depiction of ISDTs and extends their work by reference to other sources, providing for a more complete, yet arguably simpler, definition of design theories.

The paper begins by examining different perspectives on design theories. The problems with existing work are highlighted and a way forward is proposed, which recognizes an overarching set of eight components of an ISDT: (1) purpose and scope; (2) constructs; (3) principles of form and function; (4) design mutability; (5) testable propositions; (6) principles of implementation; (7) justificatory knowledge; (8) an expository instantiation. Each component is defined and discussed. The applicability of this ontological specification language is illustrated through the analysis of examples of design research. The paper concludes with a discussion of the implications of this delineation of an ISDT.

## II. APPROACHES TO DESIGN THEORIZING

A number of perspectives on design research and theory that preceded the work by Walls et al. (1992) are outlined here under the headings of the philosophy of science and technology, constructive research and design science, and the sciences of the artificial. It will be seen that this prior work does not display a clear, logical progression – rather the work has proceeded differently and under different labels in different geographic areas (especially in Europe as opposed to North

America) and in different research traditions and disciplines. Researchers in some streams of thought appear to be unaware of work that has occurred previously and there has been little prior attempt to integrate the different perspectives. The review of a number of different streams of thought gives a basis for the subsequent critical examination of the work by Walls et al. and the proposal for its extension, by integrating ideas drawn from a number of perspectives.

## *PHILOSOPHY OF SCIENCE AND TECHNOLOGY*

When talking about the nature of theory, a logical place to start is the philosophy of science, which has dealt with issues concerning theory building, specification and testing exhaustively for a very long time. In general, philosophers of science writing in the tradition of the physical or natural sciences are likely to see theory as providing **explanations** and **predictions** and as being **testable**. For example, Popper (1980) held that theorizing, in part, involves the specification of universal statements in a form that enables them to be tested against observations of what occurs in the real world. Popper described theory as follows (1980, p. 59):

> *Scientific theories are universal statements. Like all linguistic representations they are systems of signs or symbols. Theories are nets cast to catch what we call 'the world'; to rationalize, to explain and to master it. We endeavour to make the mesh even finer and finer.*

Dubin's (1978) monograph is a seminal and comprehensive source for treatment of the structural nature of theory of the type that is common in the natural and social sciences. Dubin specified the seven components of this type of theory as: (1) the units whose interactions are the subject of interest; (2) laws of interaction among

the units; (3) boundaries within which the theory is expected to hold; (4) system states within which the units interact differently; (5) propositions or truth statements about the theory; (6) empirical indicators related to the terms in the propositions; and (7) testable hypotheses incorporating empirical indicators. Dubin regarded the first five of these components as essential for theory specification, while the final two components were optional and could be included for theory testing purposes.

There is one point on which we take issue with Dubin's otherwise excellent work. Dubin followed the rather narrow position of logical positivism as expressed by Duhem (1962), who believed that physical theories should exclude explanations. Logical positivism is now largely regarded as defunct for a number of good reasons (see Magee, 1998; Passmore, 1967; Popper, 1986) and the goal of explanation is now seen as central in current conceptualizations of theory (Nagel, 1979; Popper, 1980), with a web of supportive statements and underlying explanations for the propositions that are proposed as the core of the theory. As we reject the ideas of logical positivism, we are in agreement with the more prevalent view that theory should include explanations.

Recognition that theory might relate to **technology** is not common and there may be some prejudice against it among philosophers of science (see O'Hear, 1989). As an example, a recent anthology edited by Schaff and Dusak (2003) gives an overview of work in philosophy relating to technology, but it provides little to help with the task of uncovering the nature of theory in technological disciplines. The volume includes an essay by Bunge (1979), who deals with the philosophical inputs and outputs of technology and recognizes a number of high-level, cross-disciplinary theories arising from technology, including information theory, control theory and

optimization theory. Bunge notes that the problems of the philosophy of technology include the nature of technological knowledge and its relationship to scientific knowledge, but does not explore this problem in detail.

Some relevant ideas can be traced back to Aristotle's writing on the four explanations of any "thing" in *The Four Causes* (from a translation by Hooker, 1993). The sense in which Aristotle spoke of a thing being "explained" or "caused" corresponds to its "definition" and thus is relevant to the idea of a theory for specifying artifacts. Aristotle's four causes can be applied to any artifact, such as a table. These four causes explain the artifact in terms of:

- The *causa finalis*, its final cause or end, what the table is for (eating from, placing things on);
- The *causa formalis*, its formal cause or essence, what it means to be a table (possessing a raised surface which is relatively flat supported by leg(s);
- The *causa materialis*, its material cause, what it is made from (wood);
- The *causa efficiens*, its efficient cause, who or what made the table (the carpenter).

Aristotle did not relate scientific knowledge or a theory of design to his explanation of an artifact and yet his ideas have commonalities with later work, including Simon's descriptions of the artificial. Heidegger (1993, p. 313) built on Aristotle's work in seeking to identify the essence of modern technology. Heidegger showed that Aristotle's four causes differed from one another yet belonged together in considering the nature of an artifact. Further, the coming together of the four causes in an object is an example of *poiēsis*, the arising of something from out of itself, as for example, in the bursting of a blossom into bloom.

## *THE SCIENCES OF THE ARTIFICIAL*

The classic work that gives the knowledge underlying the construction of artifacts the status of theory is Herbert Simon's *The Sciences of the Artificial* (1996) first published in 1969. Simon believed design theory was concerned with how things ought to be in order to attain goals, although the final goals of design activity might not be explicitly realized, and the designer could well proceed with a search guided by "interestingness". To Simon, an objective of design activity was the description of an artifact in terms of its organization and functioning, although he believed a theory of design might only be partly formalizable. He stressed the design of a complex artifact as a hierarchy, which could be decomposed into semi-independent components, corresponding to its many functional parts.

Simon saw the design process as generally concerned with finding a satisfactory design, rather than an optimum design, with the design process affecting the final design: "both the shape of the design and the shape and organization of the design process are essential components of a theory of design" (pp. 130-131). The design process could be informed by knowledge of the laws of natural science, both for an artifact's internal operations and its interactions with the external environment. Many artifacts are designed, however, without a full understanding of the workings of its component parts and a theory of a system design "does not depend on having an adequate microtheory of the natural laws that govern the system components. Such a microtheory might indeed be simply irrelevant" (p. 19).

Simon had a number of things to say about the design of evolving artifacts, where forecasting the likely path of events is extremely difficult. In such circumstances, he

recommended the mechanisms found in adaptive systems for dealing with change: homeostatic mechanisms that make the system relatively insensitive to the environment and retrospective adjustment to the environment's variation based on feedback. Thus, design for the future need not rely on the envisioning of remote events, but can rely on adaptive mechanisms built into the design.

## *CONSTRUCTIVE RESEARCH AND DESIGN SCIENCE*

A separate strand of work parallels work on design theory but has a different focus. In this work the emphasis is on design research as a knowledge-building activity, rather than the structural nature of the knowledge or theory that results.

European researchers have exhibited one substrand of thought. Iivari (1983) distinguished theorizing at a prescriptive level early on, using the term 'systemeering', a word coined for 'systems work' to match the Swedish word 'programmering' for programming. Iivari (1991) described this activity as "constructive" research in subsequent work. Further development of these ideas can be found in Iivari, Hirschheim and Klein (1998), Jarvinen (2001) and Kasanen et al. (1993).

North American researchers described similar perspectives under the label of a "systems development" approach to research. Nunamaker, Chen and Purdin (1990-91) provided a multi-methodological approach that included the steps of theory building (conceptual frameworks, mathematical models and methods), systems development (prototyping, product development and technology transfer), experimentation (computer simulation, field experiments and laboratory experiments) and observation (case studies, surveys and field studies). Lau (1997)

and Burstein and Gregor (1999) provide further discussion.

The software engineering community has also addressed concerns with methodological design issues. For example, Gregg et al. (2001) introduced a research methodology focused on technological innovations with stages for: (i) conceptual grounding; (ii) formal description and verification; and (iii) development to demonstrate validity. Preston and Mehandjiev (2004) give a framework for classifying intelligent design theories so as to support software-engineering design and pay some attention to "knowledge representation", which corresponds to our term "design theory" and lists its elements as "requirements, component, process and goals".

Comparable work has been promoted in IS as "design science" through the work of March and Smith (1995), who developed a framework to demonstrate the relationship, activities and outputs of design and natural science research in information technology. The design science ideas of March and Smith have enjoyed recent currency with a number of authors using or building on their ideas (Au, 2001; Ball, 2001; Hevner and March, 2003; Hevner et al., 2004).

A common element in the different sub-strands of these constructive research approaches is the emphasis on the central role of the artifact, which is seen as a vital part of the process and possibly the sole, or chief, output of the research. The construction of an artifact which is sufficiently novel is seen as a significant contribution in its own right. This view is in contrast to the Walls et al. (1992) IS design theory approach, where the artifact is constructed as a "test" of the design

theory. The question this stream of research leaves us with is whether the artifact itself, the concrete instantiation, has a place in a design theory.

### *WORK IN OTHER DISCIPLINES*

A number of disciplines apart from IS have also approached the problems of design research. Several older disciplines explicitly concerned with design, including architecture and industrial design, have a history of design-science concerns. Cross (2001) traces the desire to 'scientize' design back to the 20th Century modern movement of design, noting that the term "design science" was probably introduced in the 1960s by the inventor and radical technologist, Buckminster Fuller, who called for a design science revolution based on science, technology and rationalism.

The *design patterns* approach arose in architecture (Alexander et al., 1977) and sought to describe a particular problem within a context, the forces arising from that context and a solution that resolves those forces. Design patterns have found application in a range of disciplines as diverse as object-oriented design (Gamma, et al. 1995), systems analysis (Fernandez, 1998) and the architecture of enterprise systems (Fowler, 2003).

Further relevant work appears in management (van Aken, 2004, 2005), management accounting (Kasanen et al., 1993), accounting information systems, (David et al., 2000), art (Owen, 1997) and education (Savelson et al., 2003; Kelly, 2003). Schön (1983) linked the development of professional knowledge to "reflection–in–action".

Van Aken (2004, 2005) has addressed the problem of what prescriptive

management theory might look like and advances the idea of "technological rules", which take the form: "If you want to achieve Y in situation Z, then something like action X will help" (2004, p. 227). Although of interest for our present endeavor, the field of management is less concerned with the design of products (as in database architecture) than with methods or processes (interventions), so has some limitations in being transferred to IS, where both are of interest.

Love (2001) treats design theory from a philosophical perspective across a number of design disciplines and provides a meta-theoretical method with the aim of moving towards a simplifying paradigm of design research. This work has parallels with what we are attempting in the current paper, though it takes a wider and more abstract view of the processes and levels of design theorizing.

A theme with many of these design-based researchers is the importance of addressing problems and framing advice that is relevant to practitioners, and of a research process of iterative and reflective enquiry. This work, however, while recognizing that design theory can be generated, has with a few exceptions (for example, van Aken, 2004) little to say specifically on how it can be formulated.

## *INFORMATION SYSTEMS DESIGN THEORY (ISDT)*

The task of formally specifying design theory in IS was taken up by Walls et al. (1992), who adapted Simon's ideas for the IS context. Walls and his colleagues merged Simon's ideas with those of Dubin (1978).

Walls et al. specify the components of an ISDT as: (1) meta-requirements, the class of goals to which the theory applies; (2) meta-design, the class of artifacts

hypothesized to meet the meta-requirements; (3) design method, a description of the procedures for constructing the artifact; (4) kernel design product theories, theories from natural or social sciences that govern design requirements; (5) testable design product hypotheses, statements required to test whether the meta-design satisfies the meta-requirements; (6) kernel design process theories, theories from natural or social sciences that inform the design process; and (7) testable design process hypotheses, statements required to test whether the design method leads to an artifact which is consistent with the meta-design.

There are some questions about this specification. Two of Dubin's mandatory requirements for theory specification are missing from the Walls et al. conceptualization. There is no element that corresponds to Dubin's "units", the constructs that are present in statements of relationships in the theory. Also missing is an element that recognizes that the phenomena that are studied, in both natural-science type theory and ISDT, are systems or parts of systems. Dubin introduced the component of "system states" for this purpose, so that a theory specification depicts the various states of the system that the theory covers. Dubin gives as an example Herzberg's two-factor theory of job satisfaction (Herzberg, 1966), in which one state covered by the theory is where an individual has equal levels of satisfaction and dissatisfaction.

A further difficulty with the Walls et al. specification is what appears to be the unnecessary separation of theory components for a "design process" on top of a "design product" and the lack of clear definition of what comprises a "product" and what comprises a "process". The unnecessary duplication is highlighted in the

example given by Walls et al. of Codd's (1970) relational database theory. The kernel theory of relational algebra is shown as justification for the design method, but no kernel theory is given for the design product. In fact, it is likely that here, as with many other design theories, one single kernel theory would underlie both design product and design process.

Furthermore, it is not clear exactly the nature of the things that are addressed by the "class of goals to which the theory applies". Surely a design theory as a whole could apply to either a process or a product, and only sometimes to both. The Walls et al. (p. 43) article in fact says that one example of a widely accepted ISDT is the system development life cycle (SDLC). The SDLC is a methodology that was intended for use in developing a broad range of systems. So here the object of the design theory is itself a methodology or process. In contrast, a design theory for a product such as a word-processor could be proposed that showed the architecture and functions of the system, but not specify the means of development, as the designed product could be built satisfactorily using a number of different methods. Thus it appears that the ISDT need not mandate a design process as an essential component but rather, can itself concern a generalized process, methodology or intervention as its main object (a view congruent with Van Aken, 2004 and Carlsson, 2005).

## *CONCLUSIONS FROM PRIOR WORK*

What can we conclude from this review of prior work? First, there is not a great deal of relevant previous work to draw upon. Dubin's (1978) work on the structural nature of theory for the natural and social sciences is an obvious starting point, although following a logical positivist perspective, he omitted explanations as a component of theory. Simon's (1996) work on the sciences of the artificial stresses the importance

of extending our thinking to the science of artifacts, but it does not include a detailed examination of the nature of theory that concerns artifacts. Researchers in design science have tended not to speak of theory in relation to design knowledge at all, but have focused more on design research as an activity that results in artifact construction.

Walls et al. have drawn on both Dubin and Simon to give what is arguably the most comprehensive treatment of IS design theory structure to date, and we build on their work. However, a number of issues were identified in a critical examination of the Walls et al. specification of ISDT:

1. A lack of clarity as to what ISDTs should be concerned with, whether product or process or necessarily both;
2. The omission of the mandatory “units” (constructs) and “system states” in the adaptation of Dubin’s specification of theory components;
3. A lack of consideration of the importance of a design instantiation, as stressed in the design science literature, except as a test of a theory.
4. A possibly unnecessary distinction between kernel theories for design processes and kernel theories for design products.

Against this background, we proceed with ideas for improving the specification of ISDT.

# III. PROPOSED SPECIFICATION FOR ISDT

We have used an analytic approach in proposing a revised specification framework for ISDT, following the arguments advanced in the preceding section. An analytic approach appears appropriate for our investigation, as prior work exists that has enjoyed some recognition, albeit with some deficiencies that we have been able to identify through analysis and comparison across approaches.

Before advancing this revised framework, however, it is necessary to clarify the terminology to be used. First, it is proposed that a design theory can have as a primary design goal either (a) a methodology, such as the SDLC, or (b) a product, such as a decision support system. These design goals correspond in van Aken's terminology to "object-design" and "realization-design" (Van Aken, 2004, p. 226).

The range of artifacts that are the object of design in the discipline of IS is illustrated in publications appearing in leading IS journals. The essay on the nature of theory in IS by Gregor (2006) analysed all research articles in *MISQ* and *ISR* from March 2003 to June 2004. Nine of the 50 articles examined were classified as presenting theory for design and action. The artifacts that were the object of design theorizing included customer-centric websites, auction markets for supply chain organizations, schema for interorganizational workflows, organizational processes, and an information intermediary. This range covers both process and product artifacts, including those that are applied in organizational settings as well as more technical artifacts.

A design theory is something in an abstract world of man-made things, which also includes other abstract ideas such as algorithms and models. A design theory instantiated would have a physical existence in the real world. Figure 1 shows these different artifacts in relation to their human creators. This diagram is provided as there is sometimes confusion about the products and objects of interest in design research. March and Smith (1995) and Hevner et al. (2004, p. 78) see four design artifacts produced by design-science research: “constructs, models, methods and instantiations”. However these authors tend to see “theory” as the preserve of the natural-science type, although, on occasion, they use the word “theory” for the knowledge produced by design science. We would argue, using authorities such as Dubin (1978) and Nagel (1979) as a reference, that “constructs, models and methods” are all one type of thing and can be equated to theory or components of theory, while instantiations are a different type of thing altogether.

Our position depends on a realist ontology being adopted, where realism implies that the world contains certain types of entities that exist independently of human beings and human knowledge of them (as opposed to idealism). At a high level our ontological position corresponds to ideas expressed by both Habermas and Popper. Habermas (1984) recognizes three different worlds - the objective world of actual and possible states of affairs, the subjective world of personal experiences and beliefs, and the social world of normatively regulated social relations. These three worlds are related to Popper’s Worlds 1, 2 and 3 (Popper, 1986). World 1 is the objective world of material things, World 2 is the subjective world of mental states, and World 3 is an objectively existing but abstract world of man-made entities – language, mathematics, knowledge, science, art, ethics and institutions. Thus,

theory as an abstract entity belongs to World 3. A similar view is expressed by Love (2000) in relation to design theory. This stance is also congruent with forms of realism enjoying currency in IS and allied fields. Mingers (2000) shows how the philosophy of *critical realism* (following Bhaskar, 1989) can be applied to management science, where critical realism aims to establish a realist view of being in the ontological domain, while accepting the relativism of knowledge as socially and historically conditioned in the epistemological domain. Popper's Worlds 1 and 3 parallel the *intransitive* and *transitive* domains of Bhaskar. The relationships and interactions among these domains remain the subject of debate, yet the broad distinctions drawn here are important ones. The intransitive domain of objects and actions serves as a reference point and testing ground for theories that are the work of human beings in the transitive domain.

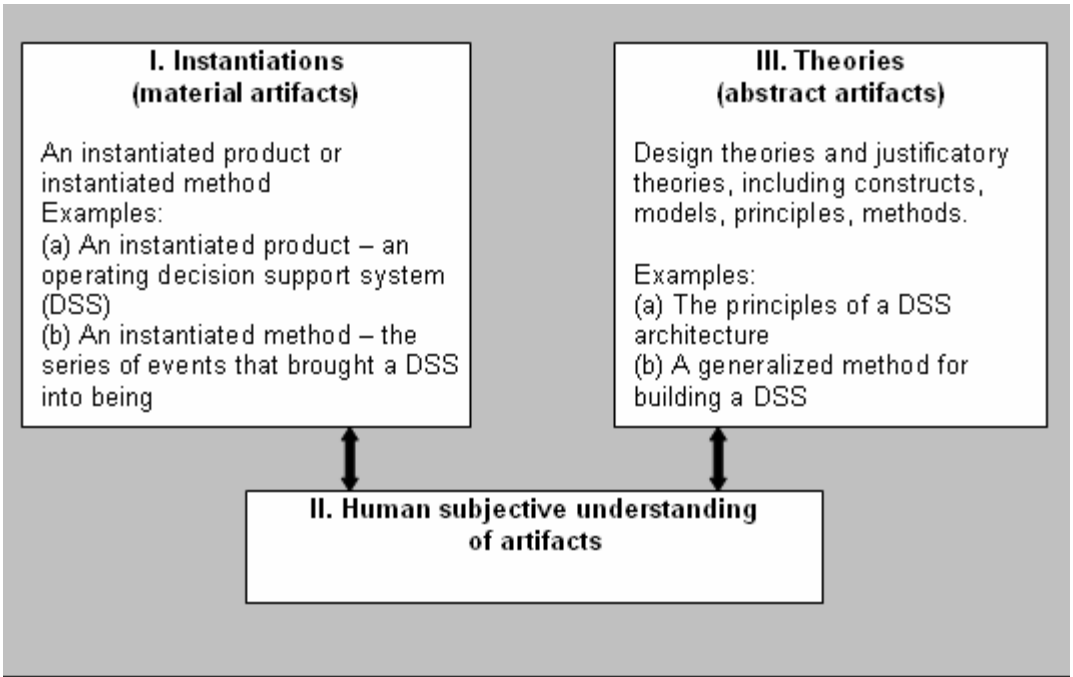

Figure 1. Relationships among IS/IT artifacts

To be more precise, the phenomena of interest for design research include:

1 *Instantiations or material artifacts*, which have a physical existence in the real world, as a piece of hardware or software, or IS, or the series of physical actions (the processes or interventions) that lead to the existence of a piece of hardware, software or an IS. This depiction of "processes" as material artifacts might be somewhat controversial, but we believe it is necessary for understanding the full range of design theories.

2 *Theories or abstract artifacts*, which do not have a physical existence, except in that they must be communicated in words, pictures, diagrams or some other means of representation. Constructs, methods and models are all this type of artifact, with the word model sometimes being used synomously with theory and constructs being one component of theories (Dubin, 1978).

3 *Human understanding of artifacts*. Human beings conceptualize and describe artifacts in abstract, general terms. The arrows in Figure 1 show that human beings create theories and constructs and use them to guide the building of instantiations in the real world and also to understand the material artifacts when in use. In addition, design principles and theory can be extracted from observation and inference from already instantiated artifacts.

To further define terms as they are used in this paper, an **IS design theory** shows the principles inherent in the design of an IS artifact that accomplishes some end, based on knowledge of both IT and human behaviour. The ISDT allows the prescription of guidelines for further artifacts of the same type. Design theories can be about artifacts that are either **products** (for example, a database) or **methods** (for example, a prototyping methodology or an IS management strategy). As the word "design" is both a noun and a verb, a theory can be about both the principles

underlying the form of the design and also about the **act** of implementing the design in the real world (an intervention).

As design theoretic knowledge is general, being applicable to all classes of cases, knowledge is needed for professionals to apply the knowledge in their own unique and specific cases, what Van Aken (2004) calls *process-design*. As a design theory can apply to either a generalized product architecture, or to a generalized method, we have the interesting situation in the latter case where we need to consider a "process for implementing the principles of a generalized process/method/intervention". We can have a theory about a methodology in terms of its general principles and also guidelines as to how it is implemented in specific circumstances.

It is important to clarify the ontological status of these artifacts of interest and also to understand the intricacies of the ways terms can be used, as this clarification has been lacking in the literature to date. We propose that the full specification of an ISDT could include eight components, as shown in Table 2. This framework extends that of Walls et al. by including the components of constructs, artifact mutability and an expository instantiation to overcome the shortcomings identified earlier. In addition, a single component for justificatory knowledge is shown instead of kernel theories for both product and process. Our argument is that any design theory should include as a minimum: (1) the purpose and scope; (2) the constructs; (3) the principles of form and function; (4) the artifact mutability; (5) testable propositions; and (6) justificatory knowledge. Five of these components have direct parallels in the five components specified by Dubin (1978) as mandatory for a natural-science-

type theory. The sixth component of justificatory knowledge needs to be added, to provide an **explanation** of why the design works. The goal of explanation is common to many current conceptualizations of theory as argued previously (Nagel, 1979; Popper, 1980).

| Table 2 Eight components of an Information Systems Design Theory | |
|---|---|
| Component | Description |
| Core components | |
| 1) Purpose and scope (the *causa finalis*) | "What the system is for", the set of meta-requirements or goals that specifies the type of artifact to which the theory applies and in conjunction also defines the scope, or boundaries, of the theory. |
| 2) Constructs (the *causa materialis*) | Representations of the entities of interest in the theory. |
| 3) Principle of form and function (the *causa formalis*) | The abstract "blueprint" or architecture that describes an IS artifact, either product or method/intervention. |
| 4) Artifact mutability | The changes in state of the artifact anticipated in the theory, that is, what degree of artifact change is encompassed by the theory. |
| 5) Testable propositions | Truth statements about the design theory. |
| 6) Justificatory knowledge | The underlying knowledge or theory from the natural or social or design sciences that gives a basis and explanation for the design (kernel theories). |
| Additional components | |
| 7) Principles of implementation (the *causa efficiens*) | A description of processes for implementing the theory (either product or method) in specific contexts. |
| 8) Expository instantiation | A physical implementation of the artifact that can assist in representing the theory both as an expository device and for purposes of testing. |

Specifying the first six components is sufficient to give the idea of an artifact that could be constructed. The construction of an instantiation as proof-of-concept and the development of specific methods for building further instantiations could come later. The credibility of the work is likely to be enhanced, however, by provision of an instantiation as a working example. Some particular innovative ideas may have merit, despite the lack of an instantiation. The history of computing shows some conceptual work on design, without instantiations or implementation principles, has been influential. For example, Vannevar Bush first wrote of a device he called a Memex early in the 1930s. His subsequent essay "As We May Think" in 1945 has had a pivotal influence in hypertext research, foreshadowing the concept of hypertext links, although Bush provided no real-life working model for his ideas or a method for building a Memex.

Table 3 shows how this specification compares with the standard for natural-science-type theory as supplied by Dubin and the prior specification of design-type theory by Walls et al. A comparison with the work of Hevner et al. is not included, as these authors did not focus specifically on the nature of design theory.

Table 3: Comparison of design theory approaches

| **Proposed anatomical skeleton** | **Dubin (1978)** | **Walls et al. (1992)** |
|---|---|---|
| 1. Purpose and scope | Boundaries | Meta-requirements |
| 2. Constructs | Units | |
| 3. Principles of form and function | Laws of interaction | Meta-description |
| 4. Artifact mutability | System states | |
| 5. Testable propositions | Propositions | Product hypotheses<br>Process hypotheses |
| 6. Justificatory knowledge | | Product kernel theories<br>Process kernel theories |

| | | |
|---|---|---|
| 7. Principles of implementation | | Design method |
| 8. Expository instantiation | Hypotheses and empirical indicators | |

The following section explains each of the eight components of an ISDT in more detail.

# IV. THE EIGHT COMPONENTS OF AN INFORMATION SYSTEMS DESIGN THEORY

Each of the eight components of a design theory is described in this section and illustrated by references to examples, including Codd's relational database design model (Table 1), a design theory for a fault threshold policy for software development projects (see Table 4), a design theory for risk management (Table 5) and some additional examples. Codd's theory was chosen as an example because it is a well-known product-type design theory and was also used as an example by Walls et al. (1992). Using it here shows that the additional components proposed for an ISDT are present in this theory. The example from Chiang and Mookerjee (2004) was chosen, because, while it has a narrow scope, it gives an example of a method-type theory which can be captured in a relatively brief description. The risk management example from Iversen, Mathiassen and Nielsen (2004) was chosen because it describes an organizational intervention process developed through action research and shows a wider conceptualization of an IS artifact than the previous two examples[2].

[2] This example was suggested by a reviewer of the paper, who believed it presented a challenge for the theory specification framework.

| Table 4: Components of a design theory for a software threshold fault policy | | |
|---|---|---|
| | *Type* | *Component examples* |
| (1) | Purpose and scope | The aim is to develop a fault threshold policy to determine when system integration occurs during a process of incremental systems development.<br>The policy is developed for homogeneous systems, where modules are similar in size and complexity and all faults take roughly the same effort to fix. The policy is appropriate for systems that can be tested frequently and at relatively low cost. The policy is designed to consider a number of project parameters (such as complexity). |
| (2) | Constructs | Examples are: incremental development, system integration, fault threshold, testing, faults detected. |
| (3) | Principles of form and function | The policy uses a derived expression to give dynamic guidelines for when system integration should occur, with (1) a region of no integration, (2) a region where integration occurs depending on a fault count, and (3) a region in which systems integration should always take place. |
| (4) | Artifact mutability | The designers consider the effects of team learning that occurs over multiple construction cycles and show how the policy will vary over a number of cycles. |
| (5) | Testable propositions | Predictions about outcomes are provided which are tested in simulation experiments. |
| (6) | Justificatory knowledge | Theory is offered relating to group coordination processes, team cognition, software development productivity, and fault growth models. |
| (7) | Principles of implementation | Not a great deal of detail is given on how to build a concrete version of this abstract policy in specific projects. An example is given where the formulae in the policy are applied to an imaginary scenario. It is stated that it might be necessary to build some randomness into the model in a real-life project and this is left for further work. |
| (8) | Expository instantiation | Examples of the policy in action are provided through simulations. |

Note: Adapted from Chiang and Mookerjee (2004) .

| Table 5: Components of a design theory for managing risk in software development |
|---|

| | *Type* | *Component examples* |
|---|---|---|
| (1) | Purpose and scope | The aim is to develop an approach for understanding and managing risk in software process improvement (SPI). |
| (2) | Constructs | For example: risk item, risky incident, resolution actions. |
| (3) | Principles of form and function | A risk framework is given to aid in the identification and categorization of risks and a process with four steps is given to show heuristics that can be used to relate identified risk areas to resolution strategies. |
| (4) | Artifact mutability | Suggestions for improving the approach are given for further work: one example is that parts of the approach could be packaged as a self-guiding computer-based system. |
| 5) | Testable propositions | It is claimed that the approach is adaptable to other organizational settings, although it is seen as a generic approach, rather than a procedure to be followed blindly (pp. 422-423). |
| (6) | Justificatory knowledge | The approach proposed is derived from other risk management approaches (other design theories). |
| (7) | Principles of implementation | It is stated that the approach requires facilitation by a facilitator experienced in risk management, SPI and running collaborative workshops. |
| (8) | Expository instantiation | Four examples of variants of the approach are given in descriptions of four iterations of an action research cycle. |

Note: Adapted from Iversen et al. (2004). This article contains two design theories, with the second being a more general approach for tailoring risk management to specific contexts. This second theory is omitted in the interests of simplicity.

## *1) THE PURPOSE AND SCOPE*

This design component says “what the system is for”. the set of meta-requirements or goals that specifies the type of system to which the theory applies and in conjunction also defines the scope, or boundaries, of the theory.

The artifact requirements should be understood in terms of the environment in which it is to operate. Heidegger (1993) used an example of a silver chalice, where in order to understand its purpose, we need to understand the religious ritual in which the chalice is to be used. Both the relational database theory and the software fault policy are described in terms of the context in which they are intended to operate. Statements in the article about the software faults policy show that it is meant to apply to certain environments, for example, where systems can be tested at relatively low cost. Codd (1982) described the need for the relational database model in the context of large databases being accessed by many people and where productivity is important.

These theory requirements are meta-requirements; they are not the requirements for one instance of a system, as would be the case if there was a need to build a single system in industry. The aim is to develop a design theory that is suited to a whole class of artifacts that are typified by these requirements. This component of the design theory is similar to the “scope” of other theory types, the area over which the theory is generalized, or what Dubin (1978) sees as defined by the “boundaries” of a theory. In defining the goals of an artifact, other goals are excluded and the boundaries of the theory are shown. For example, Codd’s relationship database theory is about the design of databases, not single file structures, which are outside the scope of his theory.

This aspect of the theory formulation allows different theories to be categorized, compared and extended. For example, a contribution to an ISDT for decision support systems would be expected to show that it filled some gap in existing ISDT,

offered an ISDT that was superior in some way to existing ISDT, or extended an existing ISDT for this type of system. This aspect thus provides guidance when it comes to evaluating a design theory. Codd (1982) compares the capabilities of the relational model with other non-relational models for database design. Chiang and Mookerjee (2004) claimed that their dynamic threshold policy is an advance on other non-dynamic policies that have similar aims but where the integration points are determined a priori. Iversen et al. (2004, p. 422) claim theoretical significance for their work in managing risks when they state that it is the first "comprehensive" process that helps software improvement teams manage risk.

### *2) CONSTRUCTS*

The representations of the entities of interest in the theory (Dubin's "units") are at the most basic level in any theory. These entities could be physical phenomena or abstract theoretical terms. Often the entities will be represented by words, such as "software fault", but mathematical symbols or parts of a diagram can also be used. Codd used the set theoretical expression of an "n-ary relation" to represent a relational database table. Iversen et al. (2004, pp. 401-402), describe the concepts of "risk items", "risk incidents", "resolution actions" and "heuristics" on which their theory builds.

As in any theory, the terms used to refer to the entities of interest should be defined as clearly as possible. A feature of design theories for information technologies is that a single construct in a theory can represent a sub-system that has its own separate design theory. One of the constructs in the fault threshold policy is "system integration". This process itself is composed of many different activities. This technique of decomposing design problems into semi-independent parts is one way

of dealing with complexity (Simon, 1996). The design of each component can then be carried out with some degree of independence of the design of others, since each will affect the others largely through its functioning and independently of the details of the mechanisms that accomplish the function. At the higher level it is not necessary for the designer to understand the detailed complexities of all the design sub-parts. The result is that the description of a construct in a design theory may be indicative, rather than detailed and complete.

### *3) PRINCIPLES OF FORM AND FUNCTION*

This component refers to the principles which define the structure, organization and functioning of the design product or design method. The shape of a design product is seen in the properties, functions, features or attributes that the product possesses when constructed. For example, a design theory for a word processor would show how an operational system should include file manipulation features, text manipulation features and so on, and how these features were interrelated. In a sense this component gives an abstract "blueprint" or architecture for the construction of an IS artifact. Similarly, the principles of a design method show in a generalized form the shape and features of the method, for example the steps in the waterfall model of the systems development life cycle. Iversen et al. (2004) describe in detail the heuristics that help software process improvement practitioners relate identified risk areas to possible resolution strategies through a four step process.

Much of the knowledge in IS textbooks that concerns application systems represents examples of this important component of design theories. An example is the depiction of the architecture of decision support systems (DSS) by McNurlin and

Sprague (2002) as including: (1) a database management system; (2) a model base management system; and (3) a dialogue generation system. It is interesting to note that this architectural description of DSS uses words for both their form (their components and how they are related) and also their functions (providing data, providing modeling tools, and interacting with users). The abstract form of a DSS is depicted in a diagram (p. 369) with boxes to represent the components and arcs showing the interaction between the parts (and the user).

In other design theories this intermingling of structural and functional properties in the architectural description also occurs. Codd describes both the form of relational tables and how they are used, in terms of access and manipulation. Chiang and Mookerjee showed how their software fault policy functions in predicting integration points. These observations highlight the importance of recognizing both form and function in the architectural component.

### *4) ARTIFACT MUTABILITY*

One component of ISDTs arises from consideration of the special nature of the IS artifact. There is increasing recognition of the mutable nature of these artifacts. That is, they are artifacts that are in an almost constant state of change. Simon (1996) spoke of evolving artifacts, where flexibility and adaptability could be enabled by feedback loops to refine design. O'Hear (1989, p. 220) writes of an "evolutionary trajectory" rather than "a design" for technologies and notes the attempt to predict the direction or outcome of a particular technological innovation in advance is bound to be uncertain. Jarvinen (2001) gives some consideration to "what happens after" in suggesting that evaluation should cover three stages: build, use and demolish (transition to new or death). Added to these ideas of the changing nature of the

artifact is Heidegger's discussion of poiēsis with respect to artifacts, the idea of the arising of something from out of itself, or emergent properties and behaviour. Supporting views for this component are also expressed by Orlikowski and Iacono (2001, p. 121):

> *We believe that the lack of theories about IT artifacts, the ways in which they emerge and evolve over time, and how they become interdependent with socio-economic contexts and practices, are key unresolved issues for our field and ones that will become even more problematic in these dynamic and innovative times.*

Specifying the degree of mutability of designed artifacts has some parallels with the specification of the states of a physical system covered by a natural science-type theory as recommended by Dubin (1978), but goes further in that it may deal not only with changes in system state, but also with changes that affect the basic form or shape of the artifact- as, for example, in allowing for a certain amount of adaptation or evolution.

Evidence of reflections on the mutability of designed artifacts can be found in the three examples presented. A primary objective of the relational database design was to allow users of databases and application programmers to remain unaffected by changes in the internal representation of data. The authors of the fault threshold policy consider the effects of learning by the project team and how the rate at which new faults arise will be reduced. Iversen et al. (2004) believe that their risk management approach can be used with benefit by different organizations, but that it may need to be adapted by adding specific risk items or resolution actions to suit the organizational context.

## *5) TESTABLE PROPOSITIONS*

An ISDT can give rise to testable propositions or hypotheses about the system or tool to be constructed. These propositions can take the general form “If a system or method that follows certain principles is instantiated then it will work, or it will be better in some way than other systems or methods”.

Walls et al. (1992) give the following reasons for having testable propositions:

- For testable design product hypotheses, there is a need to test whether the meta-design (the architectural principles) satisfies the meta-requirements;
- For testable design process hypotheses, there is a need to verify whether or not the design method (implementation principles) results in an artifact that is consistent with the meta-design (architectural principles).

Nunamaker et al. (1990-91) include as one of their five criteria for the evaluation of systems development work the need for the system to be testable against all the stated objectives and requirements. Van Aken (2004) distinguishes two forms of these design propositions. *Algorithmic* propositions are more general, typically have a quantitative format and can be tested on the basis of observations and statistical analyses. *Heuristic* propositions typically take the form “if you want to achieve Y in situation Z, then **something like** action X will help” (p. 227, emphasis added). This proposition is less general and represents a design exemplar that needs translation to a specific problem at hand to be used and tested.

The degree to which design knowledge can be expressed in general propositions remains an issue. Some degree of generality is recognized as a prerequisite for theory, even broadly defined (Gregor, 2006). The generality issue is a particular problem when design knowledge arises from artifact construction, action research and case studies, as it does in IS and many applied disciplines. Problems with

generating theory from practice and ideographic case studies have long been recognized (see, for example, Tsoukas, 1989). We concur with van Aken's view that design theory propositions can vary in their degree of generality - from claims that a design works **all** the time and in many contexts (as with an algorithm) to claims that a design proposition is only an approximation to what will work in different contexts. We recognize that this issue is worthy of further debate. However it is sufficient for our purposes to argue (following Simon), that even if design knowledge is only in part, or with difficulty, expressed in formal general terms, that this goal is still an important one for applied disciplines such as IS (see also Purao, 2002; Rossi and Sein, 2003; Vaishnavi and Kuechler, 2004/5).

Testing theoretical design propositions is demonstrated through an instantiation, by constructing a system or implementing a method, or possibly in rare cases through deductive logic (Gregg et al,. 2001; Hevner and March, 2003).

### *6) JUSTIFICATORY KNOWLEDGE*

This component provides the justificatory, explanatory knowledge that links goals, shape, processes, and materials. Some knowledge is needed of how material objects behave so as to judge their capabilities for a design. For example, the bandwidth of communication channels limits designs of e-commerce systems by placing limits on data carried within a time period. Knowledge of human cognitive capacities heavily influences principles of human-computer interaction design. Simon (1996) refered to these theories as "micro theories" and Walls et al. (1992) as "kernel theories". Walls et al. (1992) saw kernel theories as informing design products and design processes separately. Here we argue that these theories are a linking mechanism for a number, or all, of the other aspects of the design theory.

The nature, depth and degree of reliance on micro theories in ISDT is arguable. Theories might come from natural science, social science (Simon, 1996), other design theories, practitioner-in-use theories (Sarker and Lee, 2002) or evidence-based justification such as seen in medical research and action research (Van Aken, 2004). The design theory for managing risk in software development (Table 5) relies on prior practice, other design theory and action research, but not theory of the natural science type. Simon argued that it is possible to have a design theory with an incomplete understanding of the micro-theories on which it is based.

We do not have to know, or guess at all the internal structure of the system components, but only that part of it that is crucial to the abstraction in the design theory. An example is given of the first time-sharing computer systems, where only fragments of theory were available to guide initial designs. In any new discipline people often do things for which theory has no explanation and provides no foundation, and theory evolves only after practice has demonstrated that something works (Glass, 1996). Natural science explanations of how and why an artifact works may lag years behind the application of the artifact (March and Smith, 1995).

Should we settle for knowing that something works without knowing **why** it works? Venable (2006) argues that justificatory knowledge is not required as a necessary component of an ISDT. In contrast, we argue that it remains essential to include justificatory knowledge in ISDTs, although this knowledge could be incomplete. The justificatory knowledge provides an explanation of why an artifact is constructed as it is and why it works, and explanations are usually regarded as a desirable part of a

theory specification, assisting with their communicative purpose and the facilitation of human understanding (see Gregor, 2006; Nagel, 1979; Popper, 1980). Van Aken (2004, p. 228) offers a similar view - arguing that real breakthroughs in human understanding have occurred when tested technological rules can be grounded on scientific knowledge:

> *One can design an aeroplane wing on the basis of tested, technological (black-box) rules, but such wings can be designed much more efficiently on the basis of tested* ***and*** *grounded technological rules, grounded on the laws and insights of aerodynamic and mechanics.*

With information technology we have the interesting situation where some design knowledge is originally presented with an underlying justification from the behavioral sciences but this underlying justification is later either forgotten or neglected. Who now recalls that Codd's relational database theory had a behavioral science justification? One of the reasons for advancing relational database theory was that human programmers had difficulty with the complex reasoning needed to handle repeating groups of data items. It is important to remember this justification and that the normal forms of relational databases are not an end in themselves. In situations where efficiency is of prime importance a better design could use another database structure that allows repeating groups, to give faster access to data. Other similar examples can be found, for example in the paradigms of structured-programming and object-oriented programming. Human-computer interaction and web design offer further examples. Shneiderman (1998) shows how principles of interface design rest on models of human memory and cognition, which means that the designer has more, and deeper, knowledge to rely on when interpreting design guidelines in particular circumstances. In contrast, other books on web design

merely offer long lists of rather random and seemingly unconnected design guidelines, as in “Heathrow- literature” in management (Burrell, 1989).

It is difficult to envision situations where there is a complete absence of justificatory knowledge. Further, the limitations themselves can be important. For researchers, such limited knowledge provides indicators of potential fruitful areas for future research, with the phenomena that arise out of the creation of design science artifacts the targets of natural science or social science research (March and Smith, 1995). Takeda et al. (1990) express a similar view and see new knowledge generated through design activity when post-design reflection shows that the theories that motivated the design are incomplete. Justificatory knowledge also provides both researchers and practitioners with information useful in comparing competing ISDTs. All other considerations being equal, an ISDT with stronger, more complete justificatory knowledge would usually be the more appropriate choice.[3]

Examples show the range of theories relied upon in designs. Codd’s relational database design relied on knowledge from mathematical set theory, relational algebra and some understanding of the limitations of human cognition and human tendency to error. Chiang and Mookerjee (2004) build their policy of fault thresholds from knowledge of group coordination processes, team cognition, software development productivity, and fault growth models.

## *7) PRINCIPLES OF IMPLEMENTATION*

This component concerns the means by which the design is brought into being – a

[3] Just as we would more likely choose a medical treatment or drug when we understood something of why it worked, compared with the case when there was no underlying justification for its efficacy.

process involving agents and actions. Simon (1996, p. 130) believed that process and product were inextricably linked.

> *What we ordinarily call "style" may stem just as much from these decisions about the design process as from alternative emphases on the goals to be realized through the final design … both the shape of the design and organization of the design process are essential components of a theory of design.*

Several examples illustrate the nature of this component. Normalization principles are available in relational database theory to guide the database builder who is constructing a specific database. McNurlin and Sprague (2002) describe several methods for building instances of DSS, including DSS generators. They also describe how different processes can be followed to build variants of the DSS design: institutional DSS and "quick hit" DSS.

Principles can also be provided for the implementation in practice of an abstract, generic design method or development approach. To give an example, Sommerville (2001) shows the generic steps in the prototyping process as: (1) establish prototype objectives; (2) define prototype functionality; (3) develop prototype; and (4) evaluate prototype. Specific advice is also given on how to implement these general principles in practice: for example, to reduce prototyping costs and accelerate the delivery schedule, some functionality can be omitted from the prototype.

The example of the fault threshold policy illustrates further this concept of "the process of implementing a process". Chiang and Mookerjee (2004) showed how the policy varies with project parameters, including project complexity, the skills of the

project team, the development environment and the schedule flexibility. The provision of implementation principles in their design theory would mean specifying explicitly the steps a project manager would take to incorporate these parameters into the policy formulae in implementing the policy in a project. Iversen et al. (2004), give advice on how their generic approach to risk management can be used in specific contexts, suggesting particularly that an experienced facilitator is required.

### *8) EXPOSITORY INSTANTIATION*

Hevner et al. (2004) believed that "design research must produce a viable artifact in the form of a construct, model, method or instantiation". A realistic implementation contributes to the identification of potential problems in a theorized design and in demonstrating that the design is worth considering. The question that remains is whether an instantiation can be a component of a theory. Instantiated artifacts are things in the physical world, while a theory is an abstract expression of ideas about the phenomena in the physical world.

We make an argument for including an instantiation as a possible component in an ISDT for the purposes of theory representation or exposition. Theory in the natural sciences has traditionally been represented in natural language statements or in mathematical notation. A further consideration is that the artifact itself has some representational power - an artifact can assist with the communication of the design principles in a theory. To take an example, the placement of items on a computer screen can be described using screen coordinates. This process is tedious and the results are not very understandable. A copy of a screen display is more immediately comprehended and would serve better if one was illustrating some guidelines for a screen design. Similarly, a prototype system can often be used to illustrate how a

system functions, with better communicative power than a natural language description. However, if the instantiation or artifact is all that there is, rather than a theory of design, then following Cross's argument (2001) expressed previously, the level of knowledge is that of a craft-based discipline.

Both Codd (1970) and Chiang and Mookerjee (2004) used mock-ups of real systems to help explain their designs. Codd gave a simple example of the rows and columns and data elements in a relational database table and the fault threshold policy is demonstrated in a scenario with invented project attributes. Iversen et al. (2004) present examples of their risk management approach in a Danish bank as it evolved through an action research cycle.

## V. CONCLUDING REMARKS

The aim of this research essay was to delineate the possible components of a design theory for IS, providing an ontological language for the discussion of these theories. Eight separate components were distinguished: (1) the purpose and scope; (2) constructs; (3) principles of form and function; (4) artifact mutability; (5) testable propositions; (6) justificatory knowledge; (7) principles of implementation; and (8) an expository instantiation.

The essay reviewed prior work with relevance, including Simon's monograph on the sciences of the artificial, views from other disciplines and views from philosophy more generally. The work of Walls et al. (1992) provided a prior attempt at the specification of the components of a design theory for IS. We have extended this work, however, by merging ideas from the other sources that were reviewed and by

more careful examination of Dubin (1978) and Simon (1996). We have also taken some ideas from the design science field, particularly in regard to the potential importance of an instantiation of a design theory.

Two aspects of the ontology we propose are novel in terms of the structural nature of theory generally. The first is in recognizing the role of an instantiation of a design theory as an expository or representational tool. That is, an instantiation such as a prototype can be seen as serving a communicative purpose in illustrating the design principles that are embodied within it. The second is in recognizing the degree to which IS design theories deal with mutable, or changeable, artifacts. Design theories can deal with mutability in a number of ways, but it should be recognized that this is a special component of an IS design theory.

The detailed anatomy of a design theory that is presented is itself a theory, a theory that analyses and describes. Gregor (2006) suggests that this type of theory (Type 1) can be assessed by considering whether any framework developed is useful in aiding analysis, whether elements of the framework are meaningful, natural and well-defined and whether any categorization is complete and exhaustive. The degree to which the framework in this paper is useful to other researchers in analyzing and formulating theory is something that needs to be investigated through further application in practice. We have attempted to describe all constructs as clearly as possible, and our synthesis of prior work leads us to believe that our list of the structural components of design theory is fuller than any given previously. Analysis of examples of design theories using our framework identified no additional structural elements of theory that should be included. Other aspects of research

publications found (for example, research approach, motivation, evaluation and claims of significance) are not part of a theory. Nor was any over-specification in our framework found. There were no structural elements that could not be matched in design theory articles taken from leading IS journals. Only a small number of articles were analyzed, however, and further research could usefully test the framework against a larger sample.

This paper has focused on the structural components of design theory, but some consideration can also be given to how the ideas developed are used in practice. The listing of the theory components gives some guidelines to what might be included in an article or thesis that reports constructive research. It could be expected in a full, well-developed theory that all components would be present in some form. Theory that is in earlier stages of development might contain a sub-set of the components. For example, Hall, Paradice and Courtney (2003) propose a theory for learning-oriented knowledge management system that does not include an instantiation. The exemplar articles we studied included all eight theory components, although some had to be searched for.

Epistemological concerns regarding the building and testing of design theory and criteria for judging its worth have been dealt with elsewhere (see March and Smith, 1995; Hevner et al., 2004) and were not the focus of this essay. However, a number of relevant points can be deduced from our proposals in this essay.

The first is the importance of specifying the goals and scope of the theory clearly. It is this component that allows new theories to be compared with existing design

theories with similar goals and scope, providing a basis for judging whether the new theory offers a further contribution to knowledge. Researchers should review prior knowledge regarding the design of artifacts with similar goals, although the artifacts may be classified under different labels, reflecting our discipline's predilection for new names for new waves of technology. Thus, for example, work on a system described as a knowledge management system should review relevant prior work on expert systems or decision support systems.

Second, the nature of theory building for designs can be recognized. If we return to Simon's work, we find several descriptions of how the construction of an artifact can precede the knowledge of why it works. The extreme complexity of modern computer systems means that the design and building of systems is an iterative process, as recognized in software engineering methodologies (Sommerville, 2001) and the documentation of how and why a system works is likely to occur after the fact. Theory recorded after-the-fact is by no means less of a theory so long as it still satisfies the requirements of being abstract and general. That is, when reflecting on the construction of a particular system, one would need to represent the important principles underlying its construction in such a way that they are applicable to other systems yet to be constructed. A number of instantiations in multiple case studies may need to be studied before the general principles enabling them to function can be extracted (see van Aken, 2004).

Third, we can offer some observations about the degree to which design theorizing resembles what occurs in the natural sciences. Design activities include elements of creativity and imagination. Given many components for a system, all of which could

be combined in myriad ways, all theoretically sound, an experienced designer will likely employ some "art" in transforming the components into a novel and workable system. Simon's work is influenced by the notion that design is a creative activity and therefore may not be able to rely on existing theory. The question remains open as to whether "science" is an appropriate word to apply to IS design theory, given the degree of creativity involved.[4]

This paper makes contributions at several levels. Novice researchers should benefit from the depiction of the basic components of design theory, helping with the question of "What is design theory?" At a conceptual level, we have provided an advance on previous work in systematically searching for and combining differing perspectives on the components of design theory, addressing the challenge posed by Walls et al. (2004) recently to "re-examine the structure of ISDT and enhance its usability through a better structure". The outcome is a specification framework that is more complete and contains important components that were absent in earlier work. Novel aspects of the paper are the recognition that the "mutability" of IS artifacts should be reflected in theorizing about these artifacts and that instantiations can assist in communicating a design theory.

In a practical sense, this more rigorous approach to specifying design theory should assist with the development of cumulative design theory that is relevant to industry and for raising our discipline above the craft-level. Walls et al. (1992) provided a valuable start in this direction. Our new specification is more complete and some concerns with the Walls et al specification have been addressed, which we believe

[4] Although it is recognized that science in practice is also likely to involve some aspects of creativity.

will make the specification more usable. More use and understanding of the nature of theory resulting from design research should assist with more cumulative knowledge building. Design theories are more likely to be cumulative if new attempts at theorizing clearly identify the prior theory that relates to the problem area, which is identified by the "purpose and scope" component, and then build on as much relevant prior work as possible. For example, if the problem area is "how to elicit knowledge from experts" then a researcher should identify existing work that that has tackled this problem, without concern for the labels under which the work has been done, whether in artificial intelligence, expert systems or knowledge management, or in industry case studies. It is design knowledge that is of vital concern to industry and improving design theorizing should increase the relevance of our work.

The depiction of design theory in this essay may have relevance to other applied disciplines, but it also helps define what is unique about the IS discipline, namely the construction of mutable artifacts where complexity arises from the interaction of humans with information technology. Whether the anatomical framework applies to other disciplines could be a question for further research.

## REFERENCES

Alexander, C., S. Ishikawa, and M. Silverstein (1977) *A Pattern Language: Towns, Buildings, Construction*, Oxford, UK: Oxford University Press.

Au, Y. (2001) "Design Science I: The Role of Design Science in E-Commerce Research", *Communications of the AIS,* (7).

Avison, D. and T. Wood-Harper (1990) *Multiview: An Exploration in Information Systems Development*, Maidenhead, UK: McGraw-Hill.

Ball, N. (2001) “Design Science II: The Impact of Design Science on E-Commerce Research and Practice”, *Communications of the AIS*, (7).

Baskerville, R. L. and M. D. Myers (2002) “Information Systems as a Reference Discipline, *MIS Quarterly*, (26)1, pp. 1-14.

Benbasat, I. and R. Zmud (2003) “The Identity Crisis Within the IS Discipline: Defining and Communicating the Discipline’s Core Properties”, *MIS Quarterly*, (27)2, pp. 183-194.

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

Bunge, M. (1979) “Philosophical Inputs and Outputs of Technology” in G. Bugliarello and D. Doner (eds.) *The History of Philosophy and Technology,* Urbana: University of Illinois Press, pp. 262-281. (Abridged in Scharff and Dusak, 2003).

Burrell, G. (1989) “The Absence of Philosophy in Anglo-American Management Theory”, *Human Systems Management*, (8), pp. 307-312.

Burstein, F. and S. Gregor (1999) “The Systems Development or Engineering Approach to Research in Information Systems: An Action Research Perspective”, in *Proceedings of the 10th Australasian Conference on Information Systems*, B. Hope and P. Yoong, (eds.), Victoria University of Wellington, pp. 122-134.

Bush, V. (1945) “As We May Think”, *The Atlantic Monthly*, July.

Carlsson, S. (2005) “Developing Information Systems Design Knowledge: A Critical Realist Perspective”, *The Electronic Journal of Business Research Methodology*, (3)2, pp. 93-102.

Chiang, I. R. and V. S. Mookerjee (2004) “A Fault Threshold Policy to Manage Software Development Projects”, *Information Systems Research*, (15)1, pp. 3-21.

Codd, E. F. (1970) “A Relational Model of Data for Large Shared Data Banks”, *Communications of the ACM,* (13)6, pp. 377-387.

Codd, E. F. (1982) “Relational Database: A Practical Foundation For Productivity (The 1981 Turing Award Lecture)”, *Communications of the ACM*, (2)25, pp. 109-117.

Cross, N. (2001) “Design/Science/Research: Developing a Discipline”, in the *5th Asian Design Conference: International Symposium on Design Science*, Seoul, Korea: Su Jeong Dang Printing Company.

Cushing, B. E. (1990) “Frameworks, Paradigms and Scientific Research in Management Information Systems”, *Journal of Information Systems*, (2)4, pp. 38-59.

Dahlbom, B. (1996) “The New Informatics”, *Scandinavian Journal of Information Systems*, (8)2, pp. 29-47.

David, J., G. Gerard, and W. McCarthy (2000) *Design Science: Building the Future of Accounting Information Systems*, SMAP.

Davis, G. (2000) “Information Systems Conceptual Foundations: Looking Backward and Forward” in R. Baskerville, J. Stage, and J. DeGross

(eds.) *Organizational and Social Perspectives on Information Technology,* Boston: Kluwer.

Dubin, R. (1978) *Theory Building*, *revised edition*, London: Free Press.

Duhem, P. (1962) *The Aim and Structure of Physical Theory,* New York: Atheneum.

Fernandez, E. B. (1998) "Building Systems Using Analysis Patterns", *Third International Workshop on Software Architecture*, Orlando, FL: ACM.

Fowler, M. (2003) *Patterns of Enterprise Architecture*, Boston, MA: Addison-Wesley.

Gamma, E., et al. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading, MA: Addison-Wesley.

Gane, C. and T. Sarson (1979) *Structured Systems Analysis: Tools and Techniques*,, Englewood Cliffs, NJ: Prentice-Hall.

Glass, R. (1996) "The Relationship Between Theory and Practice in Software Engineering", *Communications of the ACM*, (39)11, pp. 11-13.

Godfrey-Smith, P. (2003) *Theory and Reality*, Chicago: University of Chicago Press.

Gregg, D., U. Kulkarni, and A. Vinze (2001) "Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems", *Information Systems Frontiers*, (2)3, pp. 169-183.

Gregor, S. (2002a) "A Theory of Theories in Information Systems in S. Gregor and D. Hart (eds.) *Information Systems Foundations: Building*

*the Theoretical Base*, Canberra: Australian National University, pp. 1-20.

Gregor, S. (2002b) "Design Theory in Information Systems", *Australian Journal of Information Systems*, Special Issue, pp. 14-22.

Gregor, S. (2006) "The Nature of Theory in Information Systems", *MIS Quarterly*, (3)30, pp. 611-642.

Gregor, S. and D. Jones (2004) "The Formulation of Design Theories" in Linger, H. et al.(eds.) *Constructing the Infrastructure for the Knowledge Economy: Methods and Tools, Theory and Practice*, New York: Kluwer Academic, pp. 83-93.

Habermas, J. (1984) *Theory of Communicative Action. Reason and the Rationalization of Society,* (Vol. 1), London, U.K.: Heinemann.

Hall, D., D. Paradice, and J. Courtney (2003) "Building a Theoretical Foundation for a Learning-oriented Management System", *Journal of Information Technology Theory and Application, (*2)5, pp. 63-85.

Heidegger, M. (1993) "The question concerning technology" in *Basic Writings*, San Francisco: Harper, pp. 311 - 341 (Translated from Martine Heidegger (1954) Vortrage and Aufsatze, Gunther Neske Verlag, Pfullingen, pp. 13-44.

Herzberg, F. (1966) *Work and the Nature of Man*, Cleveland: Western Publishing.

Hevner, A. and S. March (2003) "The Information Systems Research Cycle", *IEEE Computer*, (36)11, pp. 111-113.

Hevner, A. et al. (2004) “Design Science in Information Systems Research”, *MIS Quarterly*, (28)1, pp. 75-105.

Hirschheim, R. and H. K. Klein (2004) “Crisis in the Field: A Critical Reflection on the State of the Discipline”, *Journal of the Association for Information Systems*, (5)4, pp. 237-293.

Hooker, R. (1996) *Aristotle: The Four Causes- Physics II.3*, http://www.wsu.edu:8080/~dee/GREECE/4CAUSES.HTM (current Mar. 15, 2007).

Iivari, J. (1983) *Contributions to the Theoretical Foundations of Systemeering Research and the Picoco Model*, Oulu, Finland: Institute of Data processing Science- University of Oulu, Acta Univ.

Iivari, J. (2003) “Towards Information Systems as a Science of Meta-artifacts”, *Communications of the AIS*, (12)37, Nov., pp. 568-581.

Iivari, J., R. Hirschheim, and H. K. Klein (1998) “A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies”, *Information Systems Research*, (9)2, pp. 164 – 193.

Iversen, J., L. Mathiassen, and P. Nielsen (2004) “Managing Process Risk in Software Process Improvement: An Action Research Approach”, *MIS Quarterly*, (28)3, pp. 395-434.

Jarvinen, P. (2001) *On Research Methods*, Tampere, Finland: Opinpajan Kirja.

Kasanen, E., K. Lukka, and A. Siitonen (1993) “The Constructive Approach in Management Accounting Research”, *Journal of Management Accounting Research*, (5), pp. 241-264.

Kelly, A. E. (2003) “Research as Design”, *Educational Researcher*, (32), pp. 3-4.

Klein, H. K. and M. D. Myers (1999) “A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems”, *MIS Quarterly*, (23)1, March , pp. 67 – 93.

Lau, F. (1997) “A Review on the Use of Action Research in Information Systems Studies” in L. A. Liebenau and J. DeGross (eds.) *Information Systems and Qualitative Research*, London: Chapman & Hall, pp. 31-68.

Lee, A. S. (2001) “Editorial”, *MIS Quarterly,* (25:1), pp. iii-vii.

Lewin, K. (1945) “The Research Centre for Group Dynamics at Massachusetts Institute of Technology”, *Sociometry*, (8), pp. 126-135.

Love, T. (2000) “Philosophy of Design: A Meta-theoretical Structure for Design Theory”, *Design Studies*, (21), pp. 293-313.

Lukka, K. (2000) “The Key Issues of Applying the Constructive Approach to Field Research” in T. Reponen (ed.) *Management Expertise for the New Millennium: In Commemoration of the 50th Anniversary of the Turku School of Economics and Business Administration,* Turku, Finland*:* Turku School of Economics and Business Administration, pp. 113-128.

Lyytinen, K. (2002) “Designing of What? On the Ontologies of Information System Design”, *Workshop on Managing as Designing: Creating a Vocabulary for Management Education and Research*, Weatherhead School of Management, June 14-15, http://design.case.edu/2002workshop/index.html (current June 2004).

Magee, B. (1998) *Confessions of a Philosopher*, London: Phoenix.

March, S. T. and G. F. Smith (1995) “Design and Natural Science Research on Information Technology”, *Decision Support Systems*, (15), pp. 251-266.

Markus, M., L. A. Majchrzak, and L. Gasser (2002) “A Design Theory for Systems That Support Emergent Knowledge Processes”, *MIS Quarterly*, (26)3, pp. 179-212.

Markus, M. L. and D. Robey (1988) “Information Technology and Organizational Change: Causal Structure in Theory and Research”, *Management Science*, (34)5, pp. 583-598.

McNurlin, B. C. and R. H. Sprague (2002) *Information Systems Management*, *5th edition*, Upper Saddle River, NJ: Prentice Hall.

Mingers, J. (2000) “The Contribution of Critical Realism as an Underpinning Philosophy for OR/MS and Systems”, *Journal of the Operational Research Society*, 51(11), pp. 1256-1270.

Morrison, J. and J. F. George (1995) “Exploring the Software Engineering Component in MIS Research”, *Communications of the ACM*, (7)38, pp. 80-91.

Nagel, E. (1979) *The Structure of Science*, Indianapolis, IN: Hackett Publishing Co.

Nunamaker, J. F., M. Chen, and T. Purdin (1990-91) “Systems Development in Information Systems Research”, *Journal of Management Information Systems*, (7)3, pp. 89-106.

OED (Oxford English Dictionary Online), http://dictionary.oed.com (current Aug. 10, 2004).

O’Hear, A. (1989) *Introduction to the Philosophy of Science*, Oxford, UK: Clarendon Press.

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

Owen, C. (1997) “Design Research: Building the Knowledge Base”, *Journal of the Japanese Society for the Science of Design*, (5)2, pp. 36-45.

Passmore, J. (1967) “Logical Positivism” in P. Edwards (ed.) *Encyclopaedia of Philosophy (Volume V),* New York: Macmillan, pp. 52-57.

Popper, K. (1980) *The Logic of Scientific Discovery,* London: Unwin Hyman.

Popper, K. (1986) *Unended Quest an Intellectual Autobiography*, Glasgow: Fontana.

Preston, M. and N. Mehandjiev (2004) “A Framework for Classifying Intelligent Design Theories”, *Proceedings of WISER-04,* November, Newport Beach, CA.

Purao, S. (2002) “Design Research in the Technology of Information Systems: Truth or Dare”, *GSU Department of CIS Working Paper*, Atlanta: Georgia State University.

Rossi, M. and M. Sein (2003) “Design Research Workshop: A Proactive Research Approach”, *Presentation delivered at IRIS 26*, August 9 – 12, 2003, http://tiesrv.hkkk.fi/iris26/presentation/workshop_designRes.pdf (current Jan. 16, 2004).

Sarker, S. and A. Lee (2002) “Using a Positivist Case Research Methodology to Test Three Competing Theories-in-use of Business Process Reengineering”, *Journal of the AIS*, (2)7.

Savelson, R. et al. (2003) “On the Science of Education Design Studies”, *Educational Researcher*, (1)32, pp. 25-28.

Scharff, R. C. and V. Dusek (2003) *Philosophy of Technology: The Technological Condition: An Anthology*, Malden, MA: Blackwell Publishing.

Schön, D. (1983) *The Reflective Practitioner*, New York: Basic Books.

Shneiderman, B. (1998) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Reading, MA: Addison-Wesley.

Simon, H. (1981) *The Sciences of the Artificial*, *2nd edition*, Cambridge, MA: MIT Press.

Simon, H. (1996) *The Sciences of the Artificial*, 3rd edition, Cambridge, MA: MIT Press.

Sommerville, I. (2001) *Software Engineering*, New York: Addison-Wesley.

Takeda, H. et al. (1990) “Modeling Design Processes”, *AI Magazine*, (11)4, Winter, pp. 37-48.

Turban, E. and J. Aronson (2001) *Decision Support Systems and Intelligent Systems*, Upper Saddle River, NJ: Prentice-Hall.

Tsoukas, H. (1989) “The Validity of Idiographic Research Explanations”, *Academy of Management Review,* (14)4, pp. 551-561.

Vaishnavi, V. and W. Kuechler (2004/5) “Design Research in Information Systems”, http://www.isworld.org/Researchdesign/drisISworld. (current Mar., 2006).

Van Aken, J. (2004) “Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-tested and Grounded Technological Rules”, *Journal of Management Studies*, (41)2, pp. 219-246.

Van Aken, J. (2005) “Management Research as a Design Science: Articulating the Research Products of Mode 2 Knowledge Production in Management”, *British Journal of Management*, (16)1, pp. 19-36.

Venable, J. (2006) “The Role of Theory and Theorising in Design Science Research”, *First International Conference on Design Science Research in Information Systems and Technology*, Claremont, California, pp. 1-18.

Walls, J. G., G. R. Widemeyer, and O. A. El Sawy (1992) “Building an Information System Design theory for Vigilant EIS, *Information Systems Research*, (3)1, pp. 36-59.

Walls, J. G., G. R. Widmeyer, and O. A. El Sawy (2004) “Assessing Information System Design Theory in Perspective: How Useful Was Our 1992 Rendition?”, *Journal of Information Technology Theory and Practice,* (6)2, pp. 43-58.

Weber, R. (1987) “Toward a Theory of Artifacts: A Paradigmatic Base for Information Systems Research”, *Journal of Information Systems*, (1)1, Spring, pp. 3-19.

Weber, R. (1997) *Ontological Foundations of Information Systems*, Melbourne: Coopers & Lybrand.

Wyssusek, B. (2004) “Onotology and Ontologies in Information Systems Analysis and Design: A Critique”, *Tenth Americas Conference on Information Systems*, New York, New York.