A Proposal for a Formal Definition of the Design Concept

Paul Ralph and Yair Wand

Sauder School of Business
University of British Columbia
Canada
paulralph@gmail.com, yair.wand@ubc.ca

Abstract. A clear and unambiguous definition of the design concept would be useful for developing a cumulative tradition for research on design. In this article we suggest a formal definition for the concept *design* and propose a conceptual model linking concepts related to design projects. The definition of design incorporates seven elements: agent, object, environment, goals, primitives, requirements and constraints. The design project conceptual model is based on the view that projects are temporal trajectories of work systems that include human agents who work to design systems for stakeholders, and use resources and tools to accomplish this task. We demonstrate how these two suggestions can be useful by showing that 1) the definition of design can be used to classify design knowledge and 2) the conceptual model can be used to classify design approaches.

Keywords: design, information systems design, software design project, requirements, goals, science of design.

1 Introduction

There have been several calls for addressing design as an object of research. Freeman and Hart [1] call for a comprehensive, systematic research effort in the science of design: "We need an intellectually rigorous, formalized, and teachable body of knowledge about the principles underlying software-intensive systems and the processes used to create them," (p. 20). Simon [2] calls for development of a "theory of design" and gives some suggestions as to its contents (p.134). Yet, surprisingly, it seems no generally-accepted and precise definition of design as a concept is available.[1]

A clear understanding of what design means is important from three perspectives. From an instructional perspective, it seems obvious that any designer's education ought to include providing a clear notion of what design is. Furthermore, better understanding what design is will inform what knowledge such education could include.

From a research perspective, in any theoretical or empirical work in which design is a construct, a clear definition will help ensure construct validity. Furthermore, a clear understanding of the meaning of design will facilitate developing measures of

[1] As an anecdotal note – we have asked colleagues in several conferences to suggest a definition for "design" (in the software and IS context) and often the responses indicated IS academics did not have a well-defined notion of the concept.

K. Lyytinen et al. (Eds.): Design Requirements Workshop, LNBIP 14, pp. 103–136, 2009.

design-related constructs, such as design project success. Moreover, building a cumulative tradition of design research can benefit from having a well-defined, the alternative being different theories define design differently, or not defining it explicitly.

From a (software design) practitioner's perspective, a clear definition of design can help organize, share and reuse design knowledge. Such sharing can enhance software project success and software development productivity. Furthermore, understanding the elements of design would be useful in determining the issues and information that need to be considered in the process of design and in planning this process.

Given the potential value of a clear definition of design, our objective here is to suggest such a definition. We first seek to answer the question: what are the important elements of design as a phenomenon? We then seek to situate design in a network of related concepts.

We begin our discussion by making a distinction between *the science of design* and *the design science research paradigm* as elucidated by Hevner et al. [3]. In their view, design science research "builds and evaluates constructs, models, methods and instantiations" with "design intent" ([4], p. 256). In contrast, Freeman and Hart [1] call on the community to theorize and justify theories *about* design – what March and Smith [4] call "natural science intent" (p. 256). *Design science* is a research paradigm, like experimentalism. *Science of design* is a field of inquiry, like psychology. Here we seek to primarily address issues related to the science of design.

The paper is organized as follows. First, we synthesize a definition of design by applying concepts and suggestions in existing literature (§2). We then evaluate the proposed definition in Section 3. Section 4 situates our view of design in a conceptual model of software design projects. In Section 5, we demonstrate how the proposed definition of design can be applied to indexing design knowledge for reuse and by using the conceptual model of software design projects to classify design approaches. Finally, we discuss the implications of our definition of design for current themes in software design and requirements research (§6).

2 Proposing a Formal Definition of Design

2.1 Design in the Literature

We have conducted a review of existing definitions of the concept "design" in the literature. A list of definition we examined is provided in the *Appendix* (Table 9). We analyzed the definitions in three ways: first, we identified concepts that appeared common to several definitions (Table 1). We then analyzed each definition as to whether it appeared to have errors of omission or inclusion by testing them with respect to a set of examples. We have found that all definitions included errors of either kind or both. The detailed analysis is provided also in Table YY (Appendix). Finally, we have identified four main areas of disagreement among the definitions.

While most of the areas of agreement seem reasonable, a few appear problematic. First, some definitions confuse design with *good* design, adding desirability criteria to the definition, as evidenced by words like "optimally" [5] and "optimizing" [6]. Designs might be suboptimal, but we still call them designs. Second, organizing does not necessarily constitute design, for example, when someone returns books to their

Table 1. Frequency of Common Concepts in Analyzed Definitions

Concept	Frequency
Design as a *process*	11
Design as creation	11
Design as *planning*	7
Design as a *physical* activity (or as including implementation)	7
System (as the object of the design)	7
Design as being *deliberate*, or having a *purpose, goal* or *objective*	7
Design as an *activity*, or a collection of activities	7
Design as occurring in an environment (or domain/situation/context)	7
Artifact, as the object of the design	5
Needs or *requirements*	5
Design as a *human* phenomenon	5
Design as *organizing*	4
Parts, components or elements	4
Constraints or *limitations*	3
Process (as the object of design)	2
Design as *creative*	2
Optimizing	2
Design as a *mental* activity	2
Resources	2

proper shelves in a library, one is organizing the books into a pre-designed arrangement rather than actively performing a design task. Third, four definitions state or imply that design is strictly a human phenomenon. However, machines can also design objects (e.g., Bradel and Stewart [7] report on the design of processors using genetic algorithms).[2] Fourth, while many designers are surely creative, not all design need involve creativity. For example, design might involve relatively minor modifications to a previously created design.

Finally, we mention the four areas of disagreement we have identified. First, different objects of design arise: *system*, *artifact* and *process*. Second, disagreement exists concerning the scope of design: where or when a design begins and ends. Third, some definitions indicate that design is a physical activity, others a mental activity. Fourth, some disagreement concerns the outcome of design: is it a plan, an artifact, or a solution?

2.2 Suggesting a Definition of Design

In this section, we develop our proposed definition of design. First, Eekels [8] differentiates between the subject of the design and the object of design. The subject of the design is the (often human) ***agent*** that manifests the design. The ***design object*** is the thing being designed. Design outcomes such as an artifact, a system or a process that appear in some existing definitions are encompassed here by the more general term, design object.[3]

[2] Some research indicates this might also be the case for animals (see [9] and [10]).

[3] Note: often the object is called an artifact, when designed by humans. The more general term object allows (in principle) for non-human agents such as animals and computers.

Some definitions mention parts, components or elements of which the design object is, or is to be, composed. Obviously, all artificial physical things are made from other things. These other things might be given, or also are composed of components. We term the lowest level of components ***primitives***. Similarly, but perhaps less obviously, if we assume that atomic conceptual things, such as a single thought or idea, are not designed (but discovered or just *are available*), then all conceptual things that are designed are made from other conceptual things. Therefore, all design involves components, or primitives, which are, or can be, assembled or transformed to create the design object.[4] March and Smith [4] note that "Technology includes...materials, and sources of power" (p. 252). Materials and sources of power would be included in the set of primitives.

The outcome of a design effort is not necessarily the design object itself, but may be a plan for its construction, as pointed out by the definitions that characterize design as planning rather than building. The common factor here is that the agent specifies properties of the design object: sometimes as a symbolic representation, as in an architectural blueprint, sometimes as a mental representation, as in the picture in the painter's mind, and sometimes as the artifact itself, as in a hand-carved boomerang. We call the specified properties of the design object a ***specification***. More specifically, *a specification is a detailed description of a design object's structural properties*, namely, what primitives are assembled or modified and, if more than one component is used, how primitives are linked together to make the artifact. [5]

Practically speaking, a specifications document might include desired behaviors as well as structural properties. From the perspective of this paper, these desired behaviors are requirements – they are not strictly part of the specifications. The object's behavior *emerges* from the behavior of the individual components and their interactions. By behavior we mean the way the object responds to a given set of stimuli from its environment (including agents who interact with the artifact).

The specification may be purely mental, provided in a symbolic representation, presented as a physical model, or even manifested as the object itself.

Churchman [11] points out that "Design belongs to the category of behavior called teleological, i.e., "goal seeking" behavior," (p. 5). Many of the definitions we surveyed also included the concepts of goal, purpose or objective. It is possible the goal is not explicit or not well-defined. However, a design effort is always intentional. For example, a social networking web application can be designed, without having an articulated explicit goal, based only on the vague idea that it would be useful (and fun) to have an online space where people could connect. We would still say the web application was designed. On the other hand, accidental or unintentional discoveries are not really designed. Thus, ***goals*** are inherent to design insofar as a designer must

[4] What the set of available primitives is can be a relative issue. A designer might be given a set of components, or component types, where each might be in turn composed from lower level components. We consider primitives the set of component-types available to the designer, independent of whether they are natural, or the outcome of previous design. Furthermore, even if the components are not yet available, a designer might proceed assuming they will be available. The assumptions made about these components will become requirements for their design.

[5] This notion of specification agrees with that of Bourque and Dupuis ([14], p. 1-3), that design is the *activity that produces "a description of the software's internal structure"*.

have intentionality. However, this should not be interpreted as a requirement that a design goal is or can be explicitly specified and articulated.

Many definitions characterize the design process as occurring within an environment, domain, situation or context. Design involves *two different environments*: the environment of the design object, and the environment of the design agent. As pointed out by Alexander [12] "every design problem begins with an effort to achieve fitness between two entities: the form in question and its context." Clearly, the design process or activity also occurs within some ***environment***, even if that environment is difficult to characterize. March and Smith [4] mention the "organizational setting" (p. 252) and Hevner et al. [3] refer to "organizational context" (p. 77). For instance, the software created by a developer is intended to operate in a different environment than the developer. For the environment of the artifact the qualifier "organizational" is not always valid because, for some design objects, the environment does not have to be an organization (e.g. the environment of a pacemaker is a human body).

Many definitions also mention needs or requirements and limitations or constraints. The issue of requirements requires a clarification. If we interpret requirements strictly as a *formal* requirements document or as a set of mathematically expressible functions (as in [13]) the system is to perform, then requirements are not absolutely necessary. The primitive hunter who fashions a spear from a branch specified the spear's properties by creating it – without an explicit reference to formal requirements (let alone mathematically definable functions). However, in the sense that every designer expects or desires of the design object to possess certain properties or exhibit certain behaviors, ***requirements*** are inherent to design. Requirements are a major construct in requirements engineering and software design (see, for example [15] and [16]).

Similarly, all design must involve ***constraints***. Even if the design agent had infinite time and resources, physical design is still constrained by the laws of physics, virtual design by the speed and memory of the computational environment, and conceptual design by the mental faculties of the design agent. Constraints are a major construct in engineering design (see [2] and [17]). However, we note that, as for goals and requirements it is possible constraints are not stated or perceived explicitly.

The above analysis leads to the following suggestion for the definition of design (modeled in Figure 1). Table 2 further describes each concept in the definition.

Considering design as a process (depicted in Figure 2), the outcome is the specification of the design object. The goals, environment, primitives, requirements and constraints are, in principle, the inputs to the design process; however, often knowledge of these may emerge or change during the process. Nevertheless, the design process must begin with some notion of the object's intended environment, the type of object to design and some initial intentions. By initial intentions, we simply mean that design cannot be accidental – the design agent must have intentionality. Finally, if the type of design object changes significantly (e.g., from a software system to a policy manual), the existing design effort is no longer meaningful and a new design effort begins. The possibility of changing information is related to the possibility that the design process involves exploration. It also implies that the design might evolve as more information is acquired.

Design

(noun) a *specification* of an *object*, manifested by some *agent*, intended to accomplish *goals*, in a particular *environment*, using a set of *primitive components*, satisfying a set of *requirement*s, subject to some *constraints*;

(verb, transitive) to create a design, in an environment (where the designer operates)

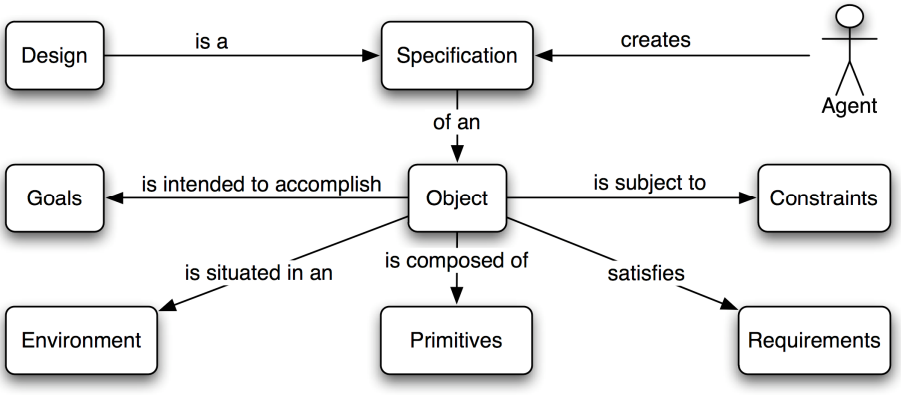

Fig. 1. Conceptual Model of Design (as a noun)

Table 2. Definitions of Design Concepts

Concept	Meaning
Design Specification	A specification is a detailed description of an object in terms of its structure, namely the components used (out of the set of possible *types* of primitives) and their connections.
Design Object	The design object is the entity (or class of entities) being designed. Note, this entity does not need to be a physical object.
Design Agent	The design agent is the entity or group of entities that specifies the structural properties of the design object.
Environment	The object environment is the context or scenario in which the object is intended to exist or operate (used for defining design as the specification of an object). The agent environment is the context or scenario in which the design agent creates the design (used for defining design as a process).
Goals	Goals are what the design object should achieve; goals are optative (i.e. indicating a wish) statements that may exist at varying levels of abstraction [18]. Since the designed object exists and/or operates in an environment, goals are related to the impact of the artifact on its environment.
Primitives	Primitives are the set of elements from which the design object may be composed (usually defined in terms of *types* of components assumed to be available).
Requirements	A requirement is a structural or behavioral property that a design object must possess. A structural property is a quality the object must posses regardless of environmental conditions or stimuli. A behavioral requirement is a required response to a *given* set of environmental conditions or stimuli. This response defines the changes that might happen in the object or the impact of these changes on its environment.
Constraints	A constraint is a structural or behavioral restriction on the design object, where "structural" and "behavioral" have the same meaning as for requirements.

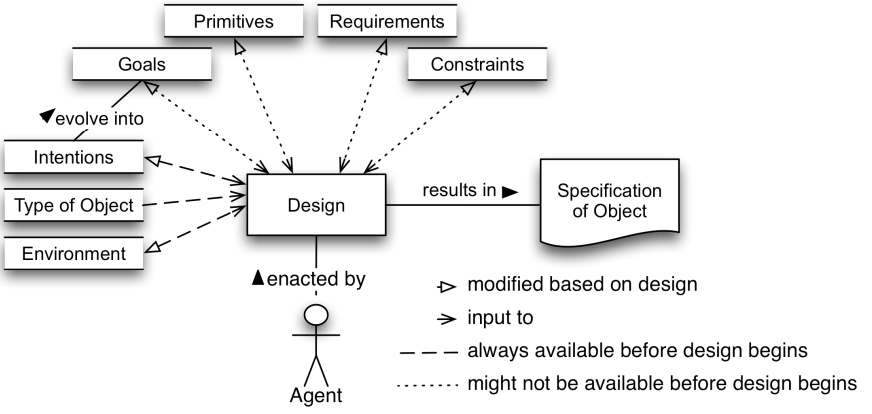

Fig. 2. Context-level Conceptual Model of Design (as a Verb)

2.3 What Can Be Designed and Examples of Design Elements

"What can be designed?" is a difficult ontological question, one we are not sure we can answer completely. However, we have identified at least six classes of design object:

- **physical artifacts,** both simple, such as boomerangs (single-component), and composite, such as houses (made of many *types* of components)
- **processes**, such as business workflows
- **symbolic systems**, such as programming languages
- **symbolic scripts**, such as essays, graphic models, and software (which, in turn, prescribe the behavior of other artifacts, i.e. computers)
- **laws, rules and policies**, such as a criminal code
- **human activity systems**, such as software development projects, committees, schools, hospitals, and artistic productions (e.g. operas)

Clearly, the nature of a specification depends on the class of design object since the structure and components of; for example, a law would be very different from those of a physical object.[6] For simple artifacts, such as a one-piece racket or a metal blade, the specification would include structural properties such as shape, size, weight and material. For a composite physical artifact, such as a desk, the specification would include the primitive components and how they are connected. Since a process is 'a set of partially ordered activities aimed at reaching a goal' [19], a specification of a process might identify the activities and their order (although other approaches are possible – e.g. using, Petri Nets [20] or states and events [21]). For a symbolic system, the specification might include syntax, denotational semantics and (for a spoken language) pragmatics. A symbolic script can be specified by symbols and their

[6] It is of interest to see how some of the concepts can be applied to non-physical artifacts such as a law. Although a law clearly is a designed (albeit conceptual) artifact, the notion of a "behavior" of a law might not be clear. One possibility would be the conditions under which it is invoked. Likewise, constraints with respect to laws can be a constitution or cultural values that limit the types of laws that can be enacted.

arrangement. A policy or law can be specified in some (possibly formal) language. The specification of a human activity system might include the various roles and tasks and their relationships and interactions.

Furthermore, all of the elements from the definition of design might vary depending on the object type. Table 3 provides examples of each design element for each type of design object.

2.4 Scope of Design

According to the perspective on design expressed in this paper, design (as a verb) is the act of specifying the structural properties of an object, either in a plan, or in the object itself. Because design is an activity, rather than a phase of some process, it may not have a discernable end point. Rather, it begins when the design agent begins specifying the properties of the object, and stops when the agent stops. Design may begin again if an agent (perhaps a user) changes structural properties of the specification or design object at a later time. This defines the scope of the design activity.

Our definition does not specify the process by which design occurs. Thus, how one interprets this scope of activities in the design process depends on the situation. If a designer encounters a problem and immediately begins forming ideas about a design object to solve the problem, design has begun with problem identification. If requirements are gathered in reaction to the design activity, design includes requirements gathering. In contrast, if a designer is given a full set of requirements upfront, or gathers requirements before conceptualizing a design object, requirements gathering is not part of design. Similarly, if the construction agent refines the specification (a possible occurrence in software development), construction is part of design, but if the designer creates a complete specification on paper that the construction agent follows, construction is not part of design. Any activity, including during testing and maintenance, that involves modifying, or occurs within an effort to modify, the specification is part of design. Therefore, design practice may not map cleanly or reliably into the phases of a particular process, such as the waterfall model [22].

This distinction has particular bearing for software design, where a significant debate over the scope of design exists. On the narrow-scope side, Bourque and Dupuis [14], for example, define design as:

> *the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software's internal structure that will serve as the basis for its construction*, (p. 3-1).

On the broad-scope side, Freeman and Hart [1], for example, argue that:

> *Design encompasses all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems—not just the activity following requirements specification and before programming, as it might be translated from a stylized software engineering process*, (p. 20).

One way of interpreting this debate is as follows. Proponents of a narrow scope of the design process posit that all inputs to design (goals, environment, primitives,

Table 3. Examples of Design Elements

Object Type	Process	Symbolic system	Law/policy	Human activity system	Physical artifact	Symbolic Script
Object	loan approval	a special purpose programming language	criminal code	a university course	office building	a software system
Agent	loan officer	person or persons who create the language	legal experts and lawmakers	instructor	Architect	programmer
Goals	accurately estimate risk level of loan	provide a means of expressing software instructions	provide a legal framework for dealing with crimes	facilitate learning and development of students in a given area	provide office space for a business	support management of customer information
Environment	bank administrative system	computing environment on which code will execute	national legal and constitutional system	university (with all resources available)	business district of a given city	personal computers and a specific operating systems
Requirements	provide a decision with justification; generate audit trail for decision process	be easily readable, minimize coder effort, fit certain applications	define crimes and punishments clearly; be unambiguous	learning objectives	include open floor plan offices, be energy efficient	maintain customer information, identify customers with certain characteristics
Primitives	various actions that need to be taken, e.g., assessing the value of a collateral	the c programming language instructions	English words as used in legal documents	various common teaching actions (presentations, laboratory sessions, tests)	building materials, interior decoration materials	the instructions in the symbolic system (programming language)
Constraints	bank approval rules and risk policies (e.g. debt-service ratio allowed)	cannot violate some programming languages related standards	must not violate the country's constitution and international laws	prior knowledge students have, number of class and laboratory hours available	comply with building code, cost less than a given budget	must be able to run on a given hardware configuration with maximum delay X.

requirements and constraints) are fully defined before any property of the object has been decided. Furthermore, the design phase results in a full specification of all relevant object properties before coding begins. In contrast, proponents of a broad scope of design recognize that properties of the object are often defined during requirements elicitation or, at the opposite end - during coding. Moreover, design might not begin with a complete knowledge of all information needed. And the process might include obtaining additional information. Which side of this debate better reflects software design practice is an empirical question; the proposed definition of design is compatible with either.

3 Evaluating the Proposed Definition of Design

In this section we evaluate our definition of design, based on the degree to which it:

- Satisfies a set of four definition evaluation criteria (Appendix, Table 8)
- Incorporates areas of agreement in existing definitions (Tables 1 and 4)
- Resolves disagreements in existing definitions (§2.1)
- Appears usable and useful.

3.1 Definition Evaluation Criteria

Coverage. Whether a definition has proper domain coverage (i.e. can account for all phenomena in the domain to which it applies) is an empirical question, akin to a universal hypothesis. Therefore, the definition cannot be proven to be correct; however, it could be shown to have coverage problems by a counter example. Thus, we evaluated the definition by testing it against a diverse set of examples (such as those in Table 3). We found that the examples could be described in terms of the seven proposed aspects of design.

Meaningfulness. This refers to the requirement that all terms used have clear meaning. We have defined explicitly all terms having imprecise everyday meanings in Table 2.

Unambiguousness. This refers to the requirement that all terms used have unique meaning. All terms with potentially ambiguous meanings are defined. All terms not explicitly defined are intended in the everyday sense, that is, as defined in the dictionary. Where terms have multiple definitions, the intention should be clear from the context.

Ease of Use. The proposed definition is presented in natural language, and is segmented into clearly distinct elements, to ensure clarity for both practitioners and researchers. It is consistent with everyday notions of design and differentiates design from related terms such as invention, decision-making, and implementation. Table 3 provides examples of the elements of design to facilitate use of the definition.

3.2 Areas of Agreement

The relationship of each area of agreement to the proposed definition is analyzed in Table 4. Aspects of design mentioned in the literature that we demonstrated should not be included are marked "discounted." As can be seen in the table, all areas are accommodated explicitly or implicitly.

Table 4. Incorporation of Areas of Agreement

Concept	Consistency with Proposed Definition
Design as a *process*	implicit in the verb form of the proposed definition
Design as creation	explicit in the verb form of the proposed definition
Design as *planning*	encapsulated by the design 'specification;' however, planning may be lightweight, especially where specification occurs simultaneously with creating the object
System (as the object of the design)	included in the more abstract term, *design object*
Design as being *deliberate*, or having a *purpose, goal* or *objective*	explicitly included as *goals*
Design as an *activity*, or a collection of activities	implicit in the verb form of the proposed definition
Design as occurring in an environment (or domain/situation/context)	explicitly included as *environment*
Artifact, as the object of the design	included in the more abstract term, *design object*
Needs or *requirements*	explicitly included as *requirements*
Design as *organizing*	Discounted
Parts, components or elements	explicitly included as *primitives*
Design as a *human* phenomenon	Discounted
Constraints or *limitations*	explicitly included as *constraints*
Process (as the object of design)	included in the more abstract term, *design object* and listed as one of the main categories of design objects
Design as *creative*	Discounted
Optimizing	Discounted
Resources	implicit in *primitives* and the verb form of the proposed definition (since creating something always uses resources)

3.3 Areas of Disagreement

The proposed definition address each of the four areas of disagreement among existing definitions described in §2.1. First, different objects of design arise: system, artifact and process. We addressed this by using the more general term, *design object* and suggesting major categories of such objects. Second, disagreement exists concerning the scope of design: where or when a design begins and ends. We discussed this issue in §2.4. Third, disagreement exists as to whether design is a physical or mental activity. Clearly, design (for humans) is mental activity, albeit one that may be supported by physical activities (such as drawing diagrams or constructing physical models). The fourth disagreement, concerning what can be designed, was addressed in §2.3.

3.4 Usefulness and Usability

We suggest that the proposed definition of the design concept can inform practice in several ways. First, the elements of the definition (excluding agent) suggest a framework for *evaluating* designs: specification – is it complete? object – did we build the right thing? goals – are they achieved? environment – can the artifact exist and operate in the specified environment? primitives – have we assumed any that are not available to the implementers? requirements – are they met, i.e., does the object possess the required properties? constraints – are they satisfied? Second, the breakdown of design into elements can provide a checklist for practitioners. Each element should be explicitly identified for a design task to be fully explicated. For example, a project team might not be able to provide consistent and accurate estimates of design project costs if crucial elements are unknown. Third, a clear understanding of design can prevent confusion between design and implementation activities. Such confusion might lead to poor decisions and evaluation practices. For example, a manager who needs to hire team members for a project might view programmers as implementers only (not understanding the design involved in programming) and thus look for the wrong sorts of skills in applicants. Fourth, the elements of design can also be used to specify and index instances of design knowledge for reuse. This is demonstrated in §4.

4 A Conceptual Model for the Design Project

We now propose a conceptual model (a set of concepts and their relationships) for design-related phenomena.[7] Here, we limit our discussion to design within the information systems field. Specifically, we view design as a *human activity that occurs within a complex entity, which can be thought of as a human activity system*. Alter [23] defines a *work system* as "a system in which human participants and/or machines perform work using information, technology, and other resources to produce products and/or services for internal or external customers," (p. 11). Expanding on this concept, we suggest that a *project* is *a temporal trajectory of a work system toward one or more goals*; the project ceases to exist when the goals are met or abandoned. Following this, we define a *design project* as a *project having the creation of a design as one of its goals*. This relationship is shown in Figure 3.

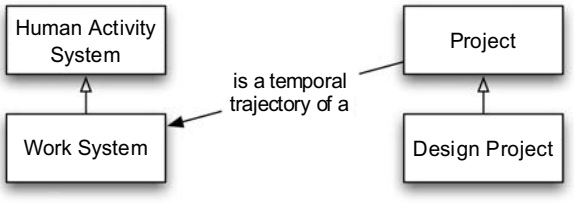

Fig. 3. Design Project Generalization Relationship. Shaded arrow indicates relationship; unshaded arrow indicates generalization.

[7] We note that to define a conceptual model of a domain, one needs to define the concepts used to reason about the domain (and their relationships). Such a conceptual structure is an ontology. Hence, we view our proposal as a conceptual model and as an ontology of concepts.

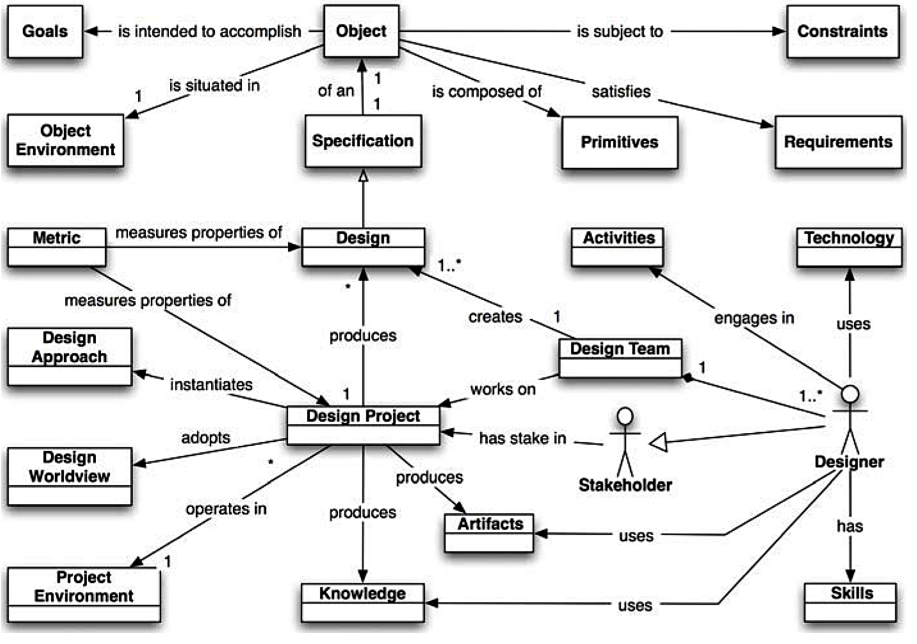

Fig. 4. Design Project Conceptual Model. Shaded arrows indicate reading direction, unshaded arrows indicate generalization, shaded diamonds indicate composition; all relationships many-to-many unless otherwise indicated.

The design project is the central concept of our conceptual model (depicted in Figure 4). Each concept is defined and each relationship is discussed in the following section (except the concepts from the definition of design, defined in Table 2).

Notes. 1) The relationships between the definition-of-design elements (e.g. constraints) and the other design project conceptual model elements (e.g., knowledge) are omitted to maintain readability. 2) The relationships between design approach and elements other than design project are unclear at this time and left for future work. 3) All shown concepts are implicitly part of the work system within which the design project takes place. 4) Creates is shown in this diagram as a relationship between design team and design, whereas Fig. 1 depicted creates as a relationship between agent and specification. In a design project, the design team is the agent. Furthermore, since the design project conceptual model includes the design concept, the model shows that the design team creates the design, which is a specification.

4.1 Discussion of Concepts

Alter [23] identifies nine elements of a work system:

- Work practices
- Participants
- Information

- Technologies
- Products and services the work system produces
- Customers for those products and services
- Environment that surrounds the work system
- Infrastructure shared with other work systems
- Strategies used by the work system and the organization

Since a design project is a trajectory of a work system, it should share all of these elements. Furthermore, since a design project is particular type of project, it should have properties not necessarily shared by other projects and work systems. Here, we discuss each element of the conceptual model, the relationships among elements, and the correspondence between elements of the conceptual model and elements of a work system. The conceptual model includes all the work system elements and, in addition, several elements specific to design projects, such as *design approach*.

Activities. Activities include the specific behaviors engaged in by participants in the design project. These may include interviewing stakeholders, modeling requirements, evaluating proposed design, etc. Activities exist at differing levels of granularity; for instance, modeling can be further divided into sub-activities such as writing scenarios, drawing entity relationship diagrams and then comparing the data models with the scenarios.

Participants and Stakeholders. Alter [23] defines participants as the "people who perform the work," (p. 13). Because individual participants vary among projects, we use the generic label, s*takeholder.* A stakeholder [24] is a person or entity with an interest in the outcome of the project. Design projects may have different types of stakeholders we specifically indicate the *designer* type for obvious reasons.

Designer. A designer is an agent that uses his or her skills to directly contribute to the creation of a design. This concept is specific to design projects.

Knowledge. Stakeholders may have and use knowledge during their involvement with the design project. In our interpretation, *knowledge* includes the kinds of information and knowhow used by stakeholders in a design project. To define knowledge, we extend the definition suggested by Bera & Wand [25]: given the states of the agent and the environment, knowledge is the information that enables an agent to select actions (from those available to the agent) so as to change the current state of affairs to a goal state. The design project can create knowledge as it proceeds – a tenant of the design science research paradigm [3].

Skill. A *skill* is a combination of mental and/or physical qualities that enable an agent to perform a specific action. Skills differ from knowledge as the latter enable one to *select* actions.

Technologies. *Technologies* are artificial, possibly intangible, tools and machines. Technologies can be used by the design team to create the design.

Design. The *design*, defined above, is the product that the design project aims to produce. This concept is specific to design projects.

Environment and Infrastructure. Fig. 4 combines Alter's *environment* and *infrastructure* constructs because both represent aspects of the project that are outside its

scope. Checkland [26] argues that, to properly model a system, the analyst must first model the system it serves. This wider system served by a design project is its environment. Alter [23] argues, "the work system should be the smallest work system that has the problems or opportunities that are being analyzed," (p. 22). Following this, then, the *environment* is the smallest coherent system served by the design project.

The environment construct is a potential source of confusion because *Design Project* and *Design* both have environments. The design project's environment is the work system in which the project occurs; the design's environment is the context in which in the object is to operate.

Design Approach and Strategy. A *design approach* is a set of beliefs about how design (and related activities) *should* be done. Examples include The Unified Software Development Process [27], and the Systems Development Lifecycle [28], [29]. According to Alter, "Strategies consist of the guiding rationale and high-level choices within which a work system, organization, or firm is designed and operates" (p. 14, [23]). As a design approach contains rationale and is implemented as choices, it corresponds to Alter's *strategy* construct. A design project may explicitly instantiate a formal design approach by using some or all of its elements. If a broad scope of design is taken (§2.4), a design approach can refer to the entire development process from problem identification to implementation and maintenance.

We have adopted the more general term, design approach, in lieu of design process or design methodology because what is referred to as "design process" often contain much more than a set of activities. Moreover, methodology is an overloaded concept used both as a formal word for 'method' and as the systematic study of methods. The design process concept is specific to design projects.

Design Team. All designers involved in a project comprise the design team. The design team engages in activities and uses technologies to create the design and other, intermediate artifacts. This concept is specific to design projects.

Artifacts. In this model, *artifact* is used in the broad, anthropological sense of any object manufactured, used or modified by agents in the design project. Examples include conceptual models, software development environments, whiteboards, and e-mails.[8]

Metric. A metric is a way or standard of taking a measurement, where measurement refers to a process of assigning symbols (often numbers) to an attribute of an object or entity (see [30], [31], [32]), and also the symbols assigned. In the case of a design project, metrics are used for evaluating specifications, designed objects, or the design project, among other things.

Design Worldview. Worldview is a way of translating the German word "Weltanschauung" meaning a way of looking onto the world. It is sometimes used in social sciences to indicate a set of high level beliefs through which an individual or group experiences and interprets the world. A precise definition of this concept is elusive. In Table 5 we suggest some possibilities for classifying Worldviews in the design context. Weltanschauungs are not mutually exclusive, i.e., a project could adopt several.

[8] This is not to be confused with the artifact that is the object of design.

Table 5. Identified Design Weltanschauung

Weltanschauung	Description	Proponents / Examples
Problem Solving	Design can be seen as an attempt to solve a known problem, a view characterized by the beliefs that a problem exists and is identifiable and that the success of a design is related to how well it solves the problem.	[2], [3], much of the design science and engineering literature.
Problem Finding	Design can be seen as an attempt to solve an unknown problem, implying that understanding the problem is part of the design process.	[33], much of the requirements engineering literature
Epistemic	Design can be seen as a learning process where actions that can lead to improvements to the current situation (in the eyes of stakeholders) are discovered.	[26]
Inspiration	Design can be seen as a result of inspiration, i.e., instead of beginning with a problem, design begins with an inspiration of the form 'wouldn't it be great if....'	the design of Facebook [34]
Growing	Design can be seen as growing an artifact, progressively improving its fit with its environment and purpose.	[4], [35]

Some design projects may explicitly adopt one or more design Weltanschauung. However, even without such an explicit view, every project participant brings a view of design to the project, and the combination of these views comprises the project's collective Weltanschauung. This concept is not necessarily common to all work systems.

4.2 Evaluation of the Conceptual Model of Design Projects

To evaluate the set of concepts underlying the proposed conceptual model, we use evaluation techniques suggested for ontologies. Ontology evaluation can proceed in several ways. The competency questions approach [36] involves simultaneously demonstrating usefulness and completeness by analytically proving that the ontology can answer each competency question in some question set. The ontology is then considered complete with respect to that question set. In contrast, Noy and Hafner [37] suggest two dimensions of ontology quality: coverage and usefulness. Coverage can be demonstrated by comparing an ontology to a reference corpus: terms in the corpus that do not fit into the ontology indicate lack of coverage. They further point out that "An important way of evaluating the capabilities and practical usefulness of an ontology is considering what practical problems it was applied to" (p. 72).

Since the proposed "ontology" is not intended to answer particular questions, evaluation with respect to coverage and usefulness seems preferable. Assessing the conceptual model's coverage is beyond the scope of this paper; however, a possible approach is evident. By surveying a range of design approaches, e.g. The Rational Unified Process, Agile Methods, The Waterfall Model, The Spiral Model, etc., A list of design concepts can be generated and compared to the proposed conceptual model.

Coverage can be measured by the extent to which these revealed concepts match the proposed concepts (usually as instances of the generic concepts suggested above).

We address usefulness in section (§5.2) by demonstrating how the conceptual model can be applied *in principle* to the practical problem of classifying and contrasting design approaches.

5 Potential Applications

In this section we discuss possible applications of the proposed definition of design and of the design project conceptual model. First, we suggest the use of the elements of the definition of design to classify and index design knowledge. Second, we discuss the use of the design project conceptual model for comparing and classifying approaches to software design.

Application 1: Design Knowledge Management System

The importance of reuse in software development has been widely recognized. For example, Mili, et al. [38] state that software reuse "is the (only) realistic opportunity to bring about the gains in productivity and quality that the software industry needs" (p. 528). Ambler [39] suggests a number of reuse types in software engineering, divided into two broad categories: *code reuse* and *knowledge reuse*.

Code reuse includes different approaches to organize actual code and incorporate it into software (e.g., libraries of modules, code fragments, or classes) and the use of off-the-shelf software. Code repositories can be considered design knowledge bases. Though some authors (e.g., [35]) argue that the best mechanism to communicate design is the code itself, sharing design is not the same as sharing design *knowledge*. Even well-commented code does not necessarily communicate design knowledge such as the rationale for structural decisions (e.g., why information was stored in a certain structure).

Knowledge reuse refers to approaches to organizing and applying knowledge about software solutions, not to organizing the solutions themselves. It includes algorithms, design patterns and analysis patterns.[9] Perhaps the most successful attempt to codify software design knowledge is the design patterns approach. A design pattern is an abstract solution to a commonly occurring problem. The design pattern concept was originally proposed in the field of architecture [40] and became popular in software engineering following the work by Gamma et al. [41].[10]

Despite the apparent benefits of sharing design knowledge, it has been observed that it is difficult to accomplish. Desouza et al. [42] claim that "Experts and veterans continue to shun reuse from public knowledge spaces" and that when the needed

[9] Other approaches to organizing software development knowledge include Architectural Patterns, Anti-Patterns, Best Practices and development methods. As well, standards and templates (e.g., for documentation) can be considered organized knowledge.

[10] The Portland Pattern Repository (http://c2.com/ppr/) is an example of a design pattern repository that could be called a design knowledge base.

artifact "was not found in their private space … it was also less costly for them to recode the desired artifact than to conduct a global search for one" (p. 98). This indicates the difficulties of locating needed design knowledge (or other software artifacts). One way to facilitate search is to classify and index design knowledge on meaningful dimensions. We now demonstrate by example how the proposed definition of design can provide such dimensions and thus help index instances of design knowledge.

An Example. In programming, an iterator object traverses a collection of elements, regardless of how the collection is implemented. Iterators are especially useful when the programmer wants to perform an operation on each element of a collection that has no index. The iterator design pattern is a description of how best to implement an iterator. Table 6 shows how the design knowledge represented by the iterator design pattern might be indexed using the elements of the proposed definition of design. Note that, in this application the goals, requirements, etc. are properties of the iterator, not of the design pattern. The goal of the design pattern, for instance, is to explain how to implement an iterator (and not to traverse a collection).

Table 6. Example of Design Knowledge Indexing

Object Type	Symbolic Script
Object	Iterator
Agent	application programmer
Goals	access the elements of a collection of objects
Environment	object-oriented programming languages
Primitives	primitives and classes available in object-oriented programming languages
Requirements	have a means of traversing a collection, be implementable with respect to a variety of collections, etc.
Constraints	must not reveal how the objects in the collection are stored, etc.

By classifying design knowledge according to these dimensions, a designer can ask questions of the form 'are there any design patterns (*object*) for traversing a collection (*requirement*) in an object-oriented language (*environment*)?' We suggest that such classification can help organize and share design knowledge and thus help improve designers' effectiveness and efficiency in locating and applying useful design knowledge.

Application 2: Design Approach Classification Framework

Classifying design approaches is important for several reasons. First, practitioners need guidance in selecting appropriate design approaches for their situations. Second, such classification can facilitate comparative research on approaches. Third, it can guide the study of the methods employed by experienced developers (which, in turn, can inform research on software design and software processes).

At least two types of classifications of design approaches are possible. First, a classification can be based on the actual elements (e.g. steps, interim products) that comprise a design approach or process. This can be termed a "white-box" approach.

Second, a classification can be based on the environment that "surrounds" a design approach. For example, specific objectives of the approach, the view of design it embeds, and the roles of stakeholders. This can be termed a "black-box" approach.

We suggest that the proposed design project conceptual model can be used to create a black-box classification scheme for design approaches. In the following we demonstrate how this can be done by examples. Using dimensions derived from the design project conceptual model, Table 8 classifies three design approaches: the Soft Systems Methodology [26], Extreme Programming [35] and the Rational Unified Process [16]. We chose these three because they are each prominent in the literature and represent significantly different perspectives.

6 Discussion and Implications for Software Design Research

6.1 Completeness, Design Agency and Software Architecture

For years, researchers have argued that informal specifications may suffer from incompleteness (e.g., [43]). Above, we defined a specification as a detailed description of an object in terms of its structure, namely the components used and their connections. This allows a more precise characterization of incompleteness. We suggest that a design specification is complete when the *relevant* structural information that has been specified is sufficient for generating (in principle) an artifact that meets the requirements.[11]

Based on the notion of completeness we have defined above, we can now identify three forms of incompleteness. First, relevant components or connections may be missing. For example, the specification for a bicycle may be missing the qualification that the tires be *attached* to the rims. Second, a particular component or connection may be insufficiently described. For example, it may not be clear from the specifications *how* the tires should be attach to the rims or which tire to use. (Please note, here we are not distinguishing here between incompleteness and ambiguity.) Third, a component may not be part of the set of primitives but can be designed based on existing primitives or other components. The design will not be complete until specifications exist for all such components.

Completeness is not an end state for a design specification. Future changes in the set of primitives may render a previously-complete specification incomplete. Furthermore, many researchers now agree on the importance of "the fluidity, or continued evolution, of design artifacts," ([44], p. 36). In situations where future conditions are difficult or impossible to predict, one response is to focus on the evolvability and adaptability of the design object [2], [45]. The characterization of design advanced here provides important implications for design fluidity. First, specification completeness does not imply constancy. A design specification can clearly be evolved over time by its original creator, the design object's users, or others, to respond to changing conditions. Furthermore, the elements of the proposed definition enumerate classes of

[11] Since it is impossible to list all of the properties of any object, we limit our discussion to "relevant" properties, i.e., a sufficient subset of properties to allow a "generating machine" (e.g., a human being or a manufacturing robot) to deterministically assemble the object.

Table 7. Example Classification of Design Approaches

	Soft Systems Methodology (SSM)	Extreme Programming	Rational Unified Process (RUP)
Object	human activity systems	software	software
Weltanschauung	epistemic	growing	problem solving
Metrics	situation dependent "measures of performance;" the 5 E's: efficacy, efficiency, effectiveness, ethicality, elegance	advocated, but none provided; differentiates internal and external quality	defines metrics as part of the process; fundamental quality measure: 'does the system do what it is supposed to?'
Nature of Specification	action items, i.e., some action that can be taken to improve the situation, in the eyes of the stakeholders	source code	UML models (use cases and diagrams); source code
Activities	semi-structured interviews, analysis, modeling, debate	coding, testing, listening, designing (refactoring)	broadly: requirements gathering, analysis and design, implementation, testing, deployment, configuration and change management, project management (each with sub activities)
Artifacts	interview guides and transcripts, collections of notes, rich pictures	prototypes, test suites	stakeholder requests, vision, business case, risk list, deployment plan, analysis model, etc.
Users	owner, actor, customer	programmers/developers, clients	RUP users take on one or more of six role categories: analysts, developers, managers, testers, production and support, and additional.
Stakeholders	stakeholders is an explicit concept in SSM	divided into "business" and "development"	"stakeholder" is a "generic role" that refers to "anyone affected by the outcome of the project" (p. 276)
Tools	rich pictures, interview guides, debates and group discussions	story cards, diagrams, an integration machine, several development workstations	IBM Rational Suite

possible changing conditions, in response to which the design object or specification might need to evolve. For example, the specification might be modified in response to changes in the environment. Finally, the set of requirements might contain stipulations for a design object's evolvability by end-users or others.

This raises questions of who exactly, in a typical software project, is the design agent? We have defined the design agent as the entity or group of entities that specifies the structural properties of the design object. When users are involved in design, whether a user is part of the design agent depends on the nature of his or her involvement. Simply providing information, such as requirements, does not make a user part of the design agent, nor does testing and giving feedback. To share in design agency, the user must *make at least one structural decision regarding the design object*. As a complete discussion of this issue would require incorporating the vast literature on authority and organizational power (e.g., [46], [47]), here we simply point out that official authority to make a structural decision does not necessarily coincide with the practical reality of who makes a decision. The key to identifying the design agent is in separating those individuals (or groups) who provide information about constraints, primitives and the other design elements, and those that decide on structural properties.

Another theme currently gaining significant attention is software architecture [44]. Software architecture is the level of design concerned with "designing and specifying the overall system structure," ([48], p.1). This presents a possible difficulty: if a specification is a description of the components of a design object and their relationships, which components and relationships are parts of the software architecture? How does one distinguish high-level components and relationships from low-level ones? A design specification for a complex system might exist simultaneously at many levels of abstraction. Alternatively (and perhaps more likely) high-level components are defined in terms of lower-level components and these are defined in terms of even lower-level components, etc., until everything is defined in terms of primitive components. In this multilevel view of design, the software architecture concept is a threshold above which is architecture, and below which is "detailed design." Is this threshold arbitrary? At this time, we can only suggest these fundamental questions about software architecture as topics for future research.

6.2 Implications for Research

The proposed characterization of design also gives rise to several implications for design research. To date, much design research has been prescriptive, addressing practical recommendations and guidance for software development; yet little theoretical, and even less empirical, treatment of software design exists [49]. This has led to many calls for field research in this area (e.g., [1], [49]). Defining design as the process by which one specifies an object's structural properties raises several important research topics:

1. How is software designed in practice?
2. To what extent is each element of the proposed definition (requirements, primitives, etc.) known when design begins?

3. Can a single theory explain all of the diverse behaviors involved in software design?
4. How do designers discover each kind of information?
5. What factors influence design project success?

Put another way, academic treatment of software design may involve developing and testing interdependent *process* and *causal* theories of design. Process theories can be used to explain *how* design occurs.[12] Causal theories deal with effects of some variables on others and can be used to suggest how to design better.

6.3 Goals Versus Requirements in Information Systems Development

The notion of *goal* is considered essential in requirements engineering as the concept that captures the motivation for developing a system ("why") and the way to define objectives at various level of abstraction [18]. Our definition of design includes both goals and requirements. We now describe briefly how these two concepts are related within this context.

We start by observing that in the information systems context, a design object is an artifact situated[13] in an environment termed the *application domain* and designed to support activities of the *application domain*. Typically, the application domain is an organizational setting such as a business or a part of a business. The application domain itself operates within an *external environment*. For example, a business is embedded within a business environment comprising customers, suppliers, competitors, service providers, and regulatory bodies. The application domain and the external environment interact: the environment generates *stimuli* that invoke *actions* in the domain. The actions of the domain can *impact* its environment. Similarly, the artifact is situated in the domain. The domain and the artifact interact: the domain creates external stimuli which invoke actions in the artifact. The actions of the artifact can impact the domain. Once the artifact is embedded a change occurs: the domain now includes the artifact. Now the modified domain (with the included artifact) interacts with the external environment. This view is depicted in Figure 5.

We define *domain goals*, or simply *goals*, as the intended impact of the actions in the domain on the external environment.[14] The purpose of the artifact is to enable the domain to accomplish these goals more effectively and efficiently. The artifact does this by responding to *stimuli* from the domain is ways that will support the domain in accomplishing the goals. Accordingly, requirements can be defined as the *properties*

[12] According to Van de Ven and Poole [50] "a process theory [is] an explanation of how and why an organizational entity changes and develops."

[13] The word "situated" should not be taken literally in the physical sense, but in sense that the artifact acts in a role of a component of the domain, and interacts with other components.

[14] To demonstrate, consider, for example, profitability, which might appear related to the business rather than to its environment. However, profitability is the outcome of exchanges between a business and its environment, and the business should act in a way these exchanges create the desired outcome.

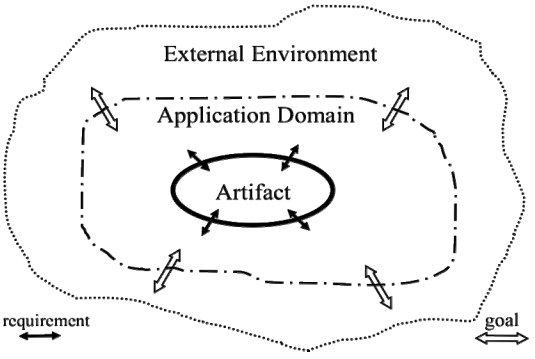

Fig. 5. Separate Domains of Goals and Requirements

that the artifact should possess in order to accomplish its purpose. These requirements can be of two types:

1. *Structural requirements* are intended to assure the artifact can match well with the other components of the domain, or those of the external environment it might interact with.
2. *Behavioral requirements* define the *desired responses* of the artifact to stimuli from the domain (or from the environment) generated when the domain is working to accomplish its goals. These responses, in turn, affect the domain (and, directly, or indirectly, the environment).

The *Requirements definition* process can be viewed as identifying what properties (structural and behavioral) the artifact should possess in order to support the domain in accomplishing the goals. Design can be viewed as the way to assemble available types of components in order to accomplish an artifact that meets the requirements.

7 Conclusion

The work we describe here is motivated by the observation that a clear, precise and generally accepted definition of the concept of design can provide benefits for research, practice and education. Our literature study indicated that such a definition was not available. We therefore undertook to propose a definition of the design concept. The definition views the design activity as a *process*, executed by an *agent*, for the purpose of generating a *specification* of an *object* based on: the *environment* in which the object will exist, the *goals* ascribed to the object, the desired structural and behavioral properties of the object (*requirements*), a given set of component types (*primitives*), and *constraints* that limit the acceptable solutions. As one possible application of our definition we demonstrate how it can be used to index design knowledge to support reuse of this knowledge.

As a second step, we situate the design concept in a network of related concepts appropriate to the information systems and software development domain by

proposing a conceptual model for design projects. The intent of this conceptual model is to facilitate study of design projects by identifying and clarifying the main relevant concepts and relationships. We demonstrate the usefulness of this conceptual model by using it to compare several approaches to system and software design.

Finally, we link our proposed definition of design to current themes in design research, in particular, the notion of requirements as used in system development.

One purpose of this work is to facilitate theoretical and empirical research on design phenomena. We hope this paper will contribute to clarifying understanding and usage of design and related concepts and encourage scientific research on design. Another purpose is to create a set of concepts that can guide practices and education in the domain of information systems and software design.

This article includes examples of design from diverse areas such as prehistoric hunters, artists, and architects. The reader may question whether such a broad perspective on design is useful for studying software development. It will be of interest to find out if software designers are more similar to engineers or to artists, or perhaps are a class on their own. This can only be answered by *observing* the behaviors of a wide range of those who are engaged in software design (elite and amateur, engineers and "hackers", formally trained and self-taught). Having a well-defined set of concepts to describe phenomena related to design and to design projects and to reason about these phenomena can provide guidance for such undertaking.

Acknowledgement. This work was done with partial support from the *Natural Sciences and Engineering Research Council of Canada*.

References

1. Freeman, P., Hart, D.: A science of design for software-intensive systems. Communications of the ACM 47(8), 19–21 (2004)
2. Simon, H.A.: The Sciences of the Artificial, 3rd edn. MIT Press, Cambridge (1996)
3. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly 28(1), 75–105 (2004)
4. March, S.T., Smith, G.F.: Design and natural science research on information technology. Decision Support Systems 15(4), 251–266 (1995)
5. Accreditation board for engineering and technology, Inc. Annual report for the year ending September 30, 1988. New York, USA (1988)
6. van Engers, T.M., Gerrits, R., Boekenoogen, M., Glassée, E., Kordelaar, P.: Power: using uml/ocl for modeling legislation - an application report. In: ICAIL 2001: Proceedings of the 8th international conference on Artificial intelligence and law, pp. 157–167. ACM Press, New York (2001)
7. Bradel, B., Stewart, K.: Exploring processor design using genetic programming. In: ECE1718 Special Topics in Computer Hardware Design: Modern and Emerging Architectures. University of Toronto, Toronto (April 2004)
8. Eekels, J.: On the fundamentals of engineering design science: The geography of engineering design science. part 1. Journal of Engineering Design 11, 377–397 (2000)
9. Breuer, T., Ndoundou-Hockemba, M., Fishlock, V.: First observation of tool use in wild gorillas. PLoS Biol. 3(11) (2005)

10. Mulcahy, N., Call, J.: Apes save tools for future use. Science 312(5776), 1038–1040 (2006)
11. Churchman, C.W.: The design of inquiring systems: Basic concepts of systems and organization. Basic Books, New York (1971)
12. Alexander, C.W.: Notes on the synthesis of form. Harvard University Press (1964)
13. Gero, J.S.: Design prototypes: A knowledge representation schema for design. AI Magazine 11(4), 26–36 (1990)
14. Bourque, P., Dupuis, R. (eds.): Guide to the software engineering body of knowledge (SWEBOK). IEEE Computer Society Press, Los Alamitos (2004)
15. Siddiqi, J., Shekaran, M.: Requirements engineering: The emerging wisdom. IEEE Software, 15–19 (March 1996)
16. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley Professional, Reading (2003)
17. Pahl, G., Beitz, W.: Engineering Design: A Systematic Approach. Springer, London (1996)
18. Lamsweerde, A.v.: Goal-oriented requirements engineering: a guided tour. In: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, pp. 249–262 (August 2001)
19. Hammer, M., Champy, J.: Reengineering the corporation: A manifesto for business revolution. Business Horizons 36(5), 90–91 (1993), http://ideas.repec.org/a/eee/bushor/v36y1993i5p90-91.html
20. van der Alast, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000)
21. Soffer, P., Wand, Y.: Goal-driven analysis of process model validity. Advanced Information Systems Engineering, 521–535 (2004)
22. Royce, W.: Managing the Development of Large Software Systems. Proceedings of IEEE WESCON 26 (1970)
23. Alter, S.: The Work system method: Connecting people, processes, and IT for business results. Work System Press (2006)
24. Freeman, R.: Strategic Management: A stakeholder approach. Pitman, Boston (1984)
25. Bera, P., Wand, Y.: Conceptual Models for Knowledge Management Systems. Working paper, University of British Columbia (2007)
26. Checkland, P.: Systems Thinking, Systems Practice. John Wiley & Sons, Ltd., Chichester (1999)
27. Jacobson, I., Booch, G., Rumbaugh, J.: The unified software development process. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
28. Department of Justice. The department of justice systems development life cycle guidance document
29. Manlei, M., Teorey, T.: Incorporating behavioral techniques into the systems development lifecycle. MIS Quarterly 13(3), 257–274 (1989)
30. Fenton, N.: Software measurement: A necessary scientific basis. IEEE Trans. Softw. Eng. 20(3), 199–206 (1994)
31. Finkelstein, L.: A review of the fundamental concepts of measurement. Measurement 2(I), 25–34 (1984)
32. Roberts, F.: Measurement Theory with Applications to Decision Making, Utility and the Social Sciences. Addison Wesley, Reading (1979)
33. Polya, G.: How to Solve It: A New Aspect of Mathematical Method, 2nd edn. Princeton University Press, Princeton (1957)

34. Kessler, A.: WSJ: Weekend interview with Facebook's Mark Zuckerberg
35. Beck, K.: Extreme programming eXplained: embrace change. Addison-Wesley, Reading (2000)
36. Grüninger, M., Fox, M.: Methodolgy for the design and evaluation of ontologies. In: Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knoweldge Sharing, Menlo Park CA, USA. AAAI Press, Menlo Park (1995)
37. Noy, N., Hafner, C.: The state of the art in ontology design. AI Magazine, 53–74 (Fall 1997)
38. Mili, H., Mili, F., Mili, A.: Reusing software: Issues and research directions. IEEE Transactions on Software Engineering 21(6), 528–562 (1995)
39. Ambler, S.: A realistic look at object-oriented reuse. Software Development 6(1), 30–38 (1998)
40. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, Oxford (1977)
41. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: Elements of reusable object-oriented software. Addison-Wesley, Boston (1995)
42. Desouza, K.C., Awazu, Y., Tiwana, A.: Four dynamics for bringing use back into software reuse. Commun. ACM 49(1), 96–100 (2006)
43. Reubenstein, H., Waters, R.: The requirements apprentice: Automated assistance for requirements acquisition. IEEE Trans. Softw. Eng. 17(3), 226–240 (1991)
44. Hansen, S., Berente, N., Lyytinen, K.: Requirements in the 21st century: Current practice and emerging trends. In: The Design Requirements Workshop, Cleveland, Ohio, USA, June 3-6 (2007)
45. Gregor, S., Jones, D.: The anatomy of a design theory. Journal of the Association for Information Systems 8, 312 (2007)
46. Aghion, P., Tirole, J.: Real and Formal Authority in Organizations. Journal of Political Economy 105, 1–29 (1997)
47. Pfeffer, J.: Managing with Power: Politics and Influence in Organizations. Harvard Business School Press, Cambridge (1992)
48. Garlan, D., Shaw, M.: An introduction to software architecture. In: Ambriola, V., Tortora, G. (eds.) Advances in software engineering and knowledge engineering, pp. 1–39. World Scientific, Singapore (1993)
49. Wynekoop, J., Russo, N.: Studying system development methodologies: an examination of research methods. Information Systems Journal 7, 47–65 (1997)
50. Van de Ven, A., Poole, M.: Explaining Development and Change in Organizations. Acad. Manage. Rev. 20, 510 (1995)
51. Hinrichs, T.R.: Problem-solving in open worlds: a case study in design. PhD thesis, Atlanta, GA, USA (1992)

Appendix: Analysis of Existing Definitions of Design

We have identified at least 33 definitions of design and sub-types of design (such as "software design" and "urban design") in the literature. Though design has several meanings, we have focused on the meaning involving plans for an object and planning or devising as a process.

We employed judgment sampling and snowball sampling, i.e., we made educated guesses as to where to look, and then investigated promising references. This strategy was consistent with our goal of identifying as many relevant definitions as possible.

To evaluate the definitions we applied a set of four main criteria: coverage, meaningfulness, unambiguousness and ease of use (see Table 8). The first three are derived from the evaluation criteria for good theories mentioned, for example, by Casti (1989, p.44-45). The fourth is a pragmatic criterion. We do not claim that these are the *best* criteria, but, in the absence of a guiding theory for evaluating definitions, that they are reasonable and have face validity.

To give the reader a sense of the thought process behind the analysis, we discuss two representative examples of the definitions encountered. The first example is by Engers et al. [6] who define design as "the creative process of coming up with a well-structured model that optimizes technological constraints, given a specification." This definition has both meaningfulness and coverage problems. First, the meaning of 'optimizes technological constraints' is unclear. In optimization techniques, one optimizes the characteristics of an object subject to constraints, not the constraints themselves. Second, the use of "well-structured" paints an idealistic portrait of design. This confounds the notion of design with measures for design quality. For example, an inexperienced computer science student can design a personal organizer application. The application might not be "well-structured", but is nonetheless *designed*. Thus, this definition omits activities that are clearly design. The second example is that of Hinrichs [51] who defines design as "the task of generating descriptions of artifacts or processes in some domain," (p. 3). This also has coverage problems. "My chair is grey" is a description of an artifact in a domain, but is clearly not a design. The problem here is that the definition relates to previously-designed artifacts. Thus, this definition includes phenomena that are not design.

The complete analysis of existing definitions is presented in Table 9. Of the 33 definitions identified, we have found that all seem to have coverage problems, at least 12 have meaningfulness problems and at least three have some form of ambiguity.

Table 8. General Definition Evaluation Criteria

	Criterion	Definition	Example of Error
Necessary	Coverage	Proper coverage means including all appropriate phenomena (completeness), and only appropriate phenomena. If a definition has improper coverage, it excludes at least one phenomenon that it should include or includes at least one phenomenon it should not.	Defining "human communication" to include only speech, will not address non-verbal communication (e.g. body language).
Necessary	Meaningfulness	Each term comprising a definition must have a commonly accepted meaning in the given context or must have been pre-defined. Each combination of terms must be directly understandable from the meaning of terms, or have been predefined.	Defining a zombie as 'the living dead' is inappropriate because, even though 'living' and 'dead' have commonly accepted meanings, their juxtaposition forms an oxymoron.
Necessary	Unambiguousness	Each term comprising a definition must have exactly one meaning in the given context; furthermore, the definition as a whole must have only one valid interpretation.	Defining political oratory as 'oral rhetoric related to politics' is inappropriate because 'rhetoric' is a contronym, i.e., has two contradictory meanings.
Optional	Ease of Use	Ideally, a definition should be easy to understand and remember, applicable in disparate situations, and readily differentiate between included and excluded phenomena. Simplicity, parsimony and concreteness are all aspects of Ease of Use. These aspects are at least in part subjective and depend on who uses the definition.	Defining the Natural Numbers as 'the smallest set satisfying the two properties: A) 1 is in N; and B) if n is in N, then n + 1 is in N" while clearly correct, would score poorly on Ease of Use in a low-level mathematics class.

Table 9. Analysis of Existing Definitions

Source	Definition	Criticism
Accreditation Board for Engineering and Technology (1988)	"Engineering design is the process of devising a system, component, or process to meet desired needs. It is a decision making process (often iterative), in which the basic sciences, mathematics, and engineering sciences are applied to convert resources optimally to meet a stated objective."	*Coverage* – the definition is idealistic and unnecessarily limiting in its use of "optimally." E.g., the building in which I work is far from optimal, but it was still designed. *Meaningfulness* – it is not clear what "desired needs" are.
Alexander (1964)	"The process of inventing physical things which display new physical order, organization, form, in response to function."	*Coverage* – this definition excludes the design of intangible things, such as processes. *Unambiguousness* – it is not clear whether thing must display new physical order, organization AND form, or new physical order, organization OR form.
Archer (1979)	"Design is, in its most general educational sense, defined as the area of human experience, skill and understanding that reflects man's concern with the appreciation and adaptation in his surroundings in the light of his material and spiritual needs."	*Coverage* – design is an activity, not an "area of human experience…" One can design with little or no experience, skill and understanding. E.g., the application programmer who designs a graphical user interface without experience in, skill in or understanding of the principles of interface design.
Beck (2000)	"Designing is creating a structure that organizes the logic in the system"	*Coverage* – excludes forms of design that organize things other than logic, e.g., urban planning organizes space.
Blumrich (1970)	"Design establishes and defines solutions to and pertinent structures for problems not solved before, or new solutions to problems which have previously been solved in a different way."	*Coverage* – Unnecessarily limits design to solutions not previously solved. Excludes independent invention and finding new ways to solve old problems. E.g., by this definition, new cars are not designed because we already have cars.
Bourque & Dupuis (2004)	"Design is defined in [IEEE610.12-90] as both "the process of defining the architecture, components, interfaces, and other characteristics of a system or component" and "the result of [that] process." Viewed as a process, software design is the software engineering life cycle activity in which software requirements are	*Coverage* – even within the software domain, this definition is far too restrictive. If someone simply writes software without creating an intermediate description of its structure, this is still design. Design is, furthermore, not limited to the phase of the software engineering life cycle between requirements analysis and construction; it is in no way clear that these phases can be practically distinguished

Table 9. (*continued*)

Source	Definition	Criticism
	analyzed in order to produce a description of the software's internal structure that will serve as the basis for its construction."	in all situations.
Buchanan (2006)	"Design is the human power to conceive, plan and realize all of the products that serve human beings in the accomplishment of their individual or collective purposes."	*Coverage* – Design is not an ability ("power") but an activity. E.g., drawing blueprints for a house, by this definition, is not design. *Unambiguousness* – it is not clear what "products" are – does this include processes and strategies as well as consumer goods?
Complin (1997)	"'design' is used to refer to the abstract description of the functional architecture of both real or possible systems."	*Coverage* – Excludes design of simple things, such as boomerangs. *Meaningfulness* – it is not clear what "functional architecture" entails
Engers et al. (2001)	"the creative process of coming up with a well–structured model that optimizes technological constraints, given a specification."	*Coverage* – excludes all suboptimal artifacts. *Meaningfulness* – the meanings of "specification" and model are unclear.
Eckroth et al. (2007)	"Design (as a verb) is a human activity resulting in a unique design (specification, description) of artifacts. Therefore, what can be designed varies greatly. However, common to all design is intention: all designs have a goal, and the goal is typically meeting needs, improving situations, or creating something new. Thus, design is the process of changing an existing environment into a desired environment by way of specifying the properties of artifacts that will constitute the desired environment; in other words, creating, modifying, or specifying how to create or alter artifacts to meet needs. In addition, it is best communicated in terms of a particular context, as previous knowledge, experience, and expectations play a strong role in designing and understanding designs."	*Coverage* – excludes independently invention of previously created artifacts and design starting from a hypothetical situations
FitzGerald and Fitz-	"design means to map out, to plan, or to arrange the	*Coverage* – this excludes artifacts that satisfy only some of their ob-

Table 9. (*continued*)

Source	Definition	Criticism
Gerald (1987)	parts into a whole which satisfies the objectives involved."	jectives. E.g., Enterprise-Resource Planning software does not always satisfy its stated objectives (Trunick 1999), but surely it as still designed.
Freeman and Hart (2004)	"design encompasses all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems—not just the activity following requirements specification and before programming, as it might be translated from a stylized software engineering process."	*Coverage* – simple systems and non-systems can also be designed, e.g. an oar, a boomerang. *Meaningfulness* – the activities are not defined or clearly explained; furthermore, enumerating the tasks *encompassed by* design does not necessarily capture the *meaning of* design.
Gero (1990)	"a goal-oriented, constrained, decision-making, exploration and learning activity which operates within a context which depends on the designer's perception of the context."	*Coverage* – The problem here is subtle. Not all design is a decision making activity; some designers, such as sculptors, may proceed fluidly without discrete decisions. It could be argued that their decisions are implicit, but then this definition would include activities such as public speaking. Decision-making is a perspective on design, not inherent to it. Furthermore, the idea of designing as leading to a new or changed artifact is missing.
Harris (1995)	"A collection of activities designed to help the analyst prepare alternative solutions to information systems problems."	*Coverage* – excludes design for non problems outside information system. *Meaningfulness* – use of "designed" is circular
Hevner et al. (2004)	"design is the purposeful organization of resources to accomplish a goal."	*Coverage* – includes organization tasks that do not constitute design, e.g., alphabetizing books. *Meaningfulness* – resources is undefined; e.g., what are the resources organized to create a military strategy? What are the resources that are being organized in graphics design? *Unambiguousness* – usage of "organization;" is it physical organization of resources, or mental?
Hinrichs (1992)	"the task of generating descriptions of artifacts or	*Coverage* – includes descriptions that are not, e.g., "the chair is

Table 9. (*continued*)

Source	Definition	Criticism
	ing, and analyzing facts about a particular [information system] and the environment in which it operates. *Systems design* then is the conception, generation and formation of a new system, using the analysis results."	a complete specification (e.g., of a bridge) rather than a system. *Meaningfulness* – this definition hinges on undefined terms "conception, generation and formation"
Jobs (2000)	"Design is the fundamental soul of a man-made creation that ends up expressing itself in successive outer layers of the product or service."	*Coverage* – excludes designs not involving a product or service and designs that are not "man-made" Meaningfulness – the meaning of "fundamental soul" is unclear
Love (2002)	"'Design'— a noun referring to a specification or plan for making a particular artefact or for undertaking a particular activity. A distinction is drawn here between a design and an artifact — a design is the basis for, and precursor to, the making of an artefact." "'Designing'—human activity leading to the production of a design."	*Coverage* – 1) the strict time sequencing implied by this definition is unnecessarily limiting; e.g., in software engineering simultaneous design and creation is arguably the preferred approach (see Martin, 1991 and Beck, 2000), 2) Design is not strictly a human activity *Meaningfulness* - "Artefact" is undefined, so the scope is unknown.
Merriam-Webster (2006) [verb]	(verb) "*transitive senses* 1 : to create, fashion, execute, or construct according to plan : DEVISE, CONTRIVE 2 a : to conceive and plan out in the mind <he *designed* the perfect crime> 4 a : to make a drawing, pattern, or sketch of b : to draw the plans for"	*Coverage* – t would include drawing a diagram of a tree (not design), but not collaboratively writing a new search algorithm (design). *Meaningfulness* – circular reference to 'design'
Merriam-Webster (2006) [noun]	"1 a : a particular purpose held in view by an individual or group <he has ambitious *designs* for his son> b : deliberate purposive planning <more by accident than *design*> 2 : a mental project or scheme in which means to an end are laid down 4 : a preliminary sketch or outline showing the main features of something to be executed : DELINEATION 5 a : an underlying scheme that governs functioning, developing, or un-	*Coverage* - Overall, this definition does not provide a unifying notion of the minimum requirements to call something a design, and does not separate designing from planning. *Meaningfulness* – circular reference to 'designs'

Table 9. (*continued*)

Source	Definition	Criticism
	complishing something (as a scientific experiment); *also* : the process of preparing this 6 : the arrangement of elements or details in a product or work of art 7 : a decorative pattern 8 : the creative art of executing aesthetic or functional designs"	
Miller's (2005)	"Design is the thought process comprising the creation of an entity,"	*Coverage* – design can encompass more than just a thought process; e.g., drawing diagrams. Thought processes cannot create physical things.
Nunamaker et al. (1991)	"Design ... involves the understanding of the studied domain, the application of relevant scientific and technical knowledge, the creation of various alternatives, and the synthesis and evaluation of proposed alternative solutions."	*Coverage* – if a person has a breakthrough idea and implements a single, innovative artifact, without considering any alternatives, this would still be design. Depending on how one defines "scientific knowledge," many designers throughout history would be excluded by this definition.
Papenek (1983)	"Design is a conscious and intuitive effort to impose meaningful order.... Design is both the underlying matrix of order and the tool that creates it."	*Coverage* – Would include all ordering activities, such as alphabetizing books *Meaningfulness* – 'underlying matrix of order' is undefined. *Ease of use* – unclear how to operationalize "matrix of order"
Partners of Pentagram (1978)	"A design is a plan to make something: something we can see or hold or walk into; something that is two-dimensional or three-dimensional, and sometimes in the time dimension. It is always something seen and sometimes something touched, and now and then by association, something heard."	*Coverage* – This definition excludes design of an incorporeal thing, e.g., a philosophy, society or strategy.
Pye (1964)	"Invention is the process of discovering a principle. Design is the process of applying that principle. The inventor discovers a class of system – a generalization – and the designer prescribes a particular embodiment	*Coverage* – Designing need not comply with principles; e.g., one might design a software interface with absolutely no knowledge of any principles regarding interface design. The interface is no less designed by someone.

Table 9. (*continued*)

Source	Definition	Criticism
Richardson (1984)	"Design is a general term, comprising all aspects of organization in the visual arts."	*Coverage* – excludes design in architecture, engineering, etc.
Schurch (1999)	"Therefore, urban design might be more clearly defined as "giving physical design direction to urban growth, conservation, and change..." (Barnett, 1982, p. 12) as practised by the allied environmental design professions of architecture, landscape architecture and urban planning and others, for that matter, such as engineers, developers, artists, grass roots groups, etc."	*Coverage* – Though design intuitively may give direction, not all instances of giving direction are design; e.g., the mere command "give the castle a moat" gives direction, but is clearly not design *Meaningfulness* – 'physical design direction' undefined
Simon (1996)	"Design is devising courses of action aimed at changing existing situations into preferred ones."	*Coverage* – excludes designs beginning from hypothetical situations, e.g., when a national defense agency designs a contingency plan for a nuclear attack, and designing imagined system.
Stumpf and Teague (2005)	"Design is a process which creates descriptions of a newly devised artifact. The product of the design process is a description which is sufficiently complete and detailed to assure that the artifact can be built."	*Coverage* – includes describing an artifact that already exists, e.g. 'the cruise ship is big;' excludes partially designed objects and design of imaginary objects.
Urban Design Group (2006)	"Urban design is the process of shaping the physical setting for life in cities, towns and villages. It is the art of making places."	*Coverage* – This definition confuses design as planning a setting with the physical process of implementing that plan; e.g., by this definition, planning the park is not designing, but laying the sods is.
Walls et al. (1992)	"The design process is analogous to the scientific method in that a design, like a theory, is a set of hypotheses and ultimately can be proven only by construction of the artifact it describes."	*Coverage* – While a design may imply a set of hypotheses, saying the design *is* the like saying being hungry *is* making a sandwich. *Ease of Use* – representing a design as a set of hypotheses may be difficult.