Equations Aren't Enough: Informal Modeling in Design

Eswaran Subrahmanian[†] Suresh L. Konda[‡] Sean N. Levy[†]
Yoram Reich[§] Arthur W. Westerberg[†]
Ira Monarch[†]

[†]Engineering Design Research Center

[‡]Software Engineering Institute
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213, USA

[§]Department of Civil and Environmental Engineering
Duke University
Durham, NC 27706, USA

In *Artificial Intelligence in Engineering Design, Analysis and Manufacturing (AI EDAM)*, 7(4):257-274, 1993

Abstract

Arguing that design is a social process, we expand the meaning of modeling and analysis to include all activities facilitating continual refinement and criticism of the design requirements, process, and solutions. We do not assume any *a priori* methods for modeling or analysis; rather, we provide a framework and an approach to study designers and give them whatever modeling and analysis capabilities they choose. Our approach is the basis for a support tool, n-dim, currently under development.

1 The Objective of Modeling and Analysis

Design as a social process involving designers, customers, and other participants consists of creating and refining a shared meaning of requirements and potential solutions through continual negotiations, discussions, clarifications, and evaluations. This shared meaning, crystalized as the design artifact and made persistent as shared memory forms the basis of accumulated experience upon which subsequent designs draw. Therefore, design requires support for the following activities: negotiating to establish shared meaning, maintaining and refining the components of the shared meaning, and maintaining and accessing prior information constituting fragments of shared memory. All these requirements are facilitated through iterative modeling and analysis (MA) activities of various forms. If the information about these MA activities is maintained properly, the development of shared meanings can be incremental. Therefore, MA activities can rely on previous experience, instead of being re-invented each time, and pitfalls typically encountered in MA can be avoided.

In the process of reaching this shared meaning, both modeling and analysis take place, albeit often in an informal and inchoate fashion. For instance, when two designers interact, their exchange involves a particular aspect of the design that is modeled in their discussion. A question posed by one designer constitutes modeling and the response an analysis. Often, the focus of the discussion or negotiation drifts marking the use of several models which, while possibly loosely connected, are nevertheless invaluable for the negotiation. Therefore, to benefit from past models arising in collaborative processes, the information derived from previous negotiations between designers needs to be maintained.

Access to information from previous, analogically related, design situations is a basic requirement for improving design. In fact, the very act of accessing and applying previous information implies a model of past information and requires models and analyses of the present. To illustrate, if designers create a query to retrieve parts from a database for satisfying a specific function, they model the functionality required using a relatively small set of parameters related to, and perhaps derived from, past models. If the query retrieves useful parts, the analysis was successful and the modeling appropriate. If the query fails, knowledge about the failure constitutes valuable information as well. Consequently, it is necessary that not only successes but also that failures be maintained.

MA activities manifest in negotiation and information retrieval are by and large *informal*, as opposed to *formal modeling* via models cast in mathematical form as traditionally conceived of in engineering.

1.1 Formal Models

Even within the sphere of formal models, a considerable degree of informal MA takes place. To begin with, all aspects of the evaluation of formal models is done, in the main, using the criterion of *sufficiency*, as in "it is sufficiently accurate," "it is a good enough model," etc. Such a criterion is, on the very face of it, not a formalizable criterion.[1] In fact, a strong argument against formalizing such a criterion can be made, since the knowledge that would necessarily be needed to formalize it is constantly changing, and quite often is in an inchoate, unstable, or even inaccessible state (e.g., in the heads of multiple people, with their personal and often conflicting perspectives, etc.). In short, the very definition of sufficiency is a negotiated outcome of the design process, and not an input, *a priori* or otherwise.

In the process of MA, formal models are evaluated along several dimensions: accuracy, applicability, intent, and mutual consistency. Observe that this is, itself, an informal model of the applicability of formal models, and is not presented here as some *a priori* truth about all formal models or methods. We note in passing that the criterion of cost is embedded within each of these dimensions; that is, cost is itself a multi-dimensional criterion (the cost of building/applying a model, the cost of a mistake due to a modeling error, etc.). Briefly, we define these dimensions below.

Accuracy: Notions of both how accurate the model is *vis a vis* some standard and how to interpret this accuracy. Often such standards are determined on benchmark problems that are only models with restricted scope or applicability. Therefore, a model may be accurate under certain conditions, for certain purposes, etc. and never be definitely accurate (note "how accurate are the results" vs. "how accurate a portrayal of the real world is the model").

Applicability: How applicable is the model to the given situation; this is related to, but not the same as accuracy. That is, will the results one gets from the model actually answer any of the questions one has, or bring up any questions one is interested in discovering?

Intent: Does the model do what one really wants it to do? That is, (a) does one have a good enough description of the "real world" (the object being designed, its context, etc.) to know that the model is fulfilling the purpose one thinks it should? and (b) does one understand the "purpose one thinks the model should fulfill" well enough to reconcile the model with the "real world"?

Mutual consistency: The union of all formal models used to describe an artifact or situation is not necessarily (a) meaningful or (b) complete. First, different models can overlap in inconsistent ways, can present conflicting results and descriptions of overlapping or similar

[1] Cf. DeMillo, Lipton, and Perlis (1979) in the domain of mathematical proofs and formal verification of computer programs.

things, and can view the world in fundamentally inconsistent ways, etc.. Second, one cannot know that all these models, taken together, describe the *whole* picture (in fact, one cannot even describe what the *whole* picture might look like).

In short, whatever the status of formal (analytical) methods and models might be in an academic setting, their use in design necessarily entails their evaluation along (at least) these four dimensions using the criterion of sufficiency, all of which is necessarily an informal process. Perhaps more to the point, this evaluation occurs in a highly context-sensitive form: whether the designer is the lone designer or a member of a design team, the evaluation of a given formal method is colored and tempered by the stage in the design cycle, the artifact being designed, the previous experiences of the designer and the team, and so forth. That is, model selection, model application, and interpretation of model results are the outcomes of a negotiation.[2]

1.2 Informal Models

From studies of design, several interesting features of how designers work in various organizations can be discerned:

- Different designers use different vocabularies to describe the same or very closely related sets of things (Sargent et al, 1992).
- Engineers typically spend at most 15% (Hales, 1987) of their time doing analytical tasks, the rest of their time being spent *negotiating* various aspects of the design, including the structure of the task of doing the design itself. This negotiation most often takes the forms of one-on-one meetings and paper being passed about within the organization.
- Since individuals tend to organize information in idiosyncratic ways, there is usually substantial overhead in the process of *merging* all of the individual representations in a design team into a single, coherent view.

All of these are informal aspects of modeling in design. Furthermore, our comments on formal models extends to all techniques (models) use by engineers including formal or informal, ad-hoc, experimental, verbal (peer feedback), qualitative or quantitative, precise or approximate. The key to engineering practice appears to be a simple pragmatism; anything that works, goes (Feyerabend, 1975; Konda et al, 1992; Petroski, 1992). Engineers also use

[2]In fact, this is true even in the academic setting where different modeling techniques are presented for peer scrutiny and some become more acceptable than others, not always due to "objective" evaluation.

a variety of media and modes. The media run the gamut from sketches, notes, diagrams, abstract graphical objects, initially loosely organized with fragmented content. The modes range from exchanges in absentia to gestures and facial expressions (Leifer, 1991). The practice of design which involves multiple representations, disciplines, and designers, introduces the need to broaden the horizon of modeling techniques. This is not just being in line with present interest in concurrent engineering but rather the acknowledgment that a single entity misses crucial contributions to design provided by peers and other affected parties.

The connecting thread between these activities or approaches is that they are all expected to provide insight into the problem at hand; they are meant to facilitate a better understanding of needs, problems encountered, and potential solutions. In this expanded view of design, therefore, MA are *any and all* activities facilitating *understanding*. An "ideal" understanding facilitates efficient problem solving since understanding *requires* shared meaning which is, among other things, a negotiated common vocabulary (Konda et al, 1992). However, such a state is not necessarily attainable, but can be approached via modeling, model utilization (analysis), and model refinement in the dialectic of negotiation.

The resulting models of the design process, the design requirements, the design solutions (however approximate, formal, or informal) can be codified into modeling conventions.[3] These conventions congeal in a social context, and their future utility is determined in yet another social context. The conventions are, as a consequence, necessarily results of both socio-linguistic and more formal, precise, albeit limited, formal languages. Perhaps some consolation can be derived by pure formalists from the observation that in the purest of intellectual pursuits — mathematics — proofs are determined to be correct and valid as social constructions (DeMillo et al, 1979; Kanigel, 1991).

1.3 Combining informal and formal modeling

We have seen that formal modeling techniques are also inherently informal, not in their formulation, but in their applicability to practical design. Therefore, although most design activities, especially those dealing with complex systems, end with detailed MA using a variety of formal methods, designers necessarily combine both types of MA in their work. This paper elaborates on this property of design and then presents a tool, *n*-dim, that supports informal as well as formal MA activities. The remainder of this paper is organized as follows. Section 2 provides two case studies on the use of formal models in engineering. Sections 3 and 4 discuss several observations on MA and outline some requirements for systems that are intended to support these activities in actual practice. Section 5 introduces

[3] These conventions are operationalized by what are called *modeling languages* in *n*-dim. See Section 5 for a more detailed discussion.

n-dim, a system built upon the guidelines presented. Section 6 illustrates the use of n-dim for both formal and informal modeling. Section 7 concludes the paper.

2 Case Studies in the Utility of Formal Models in Design

In this section, we present data on MA in two engineering situations: a basic engineering problem (behavior of plastic materials) and a more comprehensive engineering problem (the design of ship hulls). We emphasize the incompleteness of formal MA in these engineering practices, thus leading to the need for informal MA.

2.1 Basic engineering models

Often, it may be perceived that a compilation of information about possible models yields comprehensible and comprehensive knowledge. For example, Table 1 contains a comparison of the utility of different equational models for describing the behavior of plastic materials. Such equations exemplify the most fundamental kind of engineering knowledge, based on experience, or on theoretical models of behavior and their calibrations through experimental testing. While the representation of these equations is formal, their development through theorizing and negotiation and their experimental testing are themselves based on models with intrinsic, often tacit, underlying assumptions.

Table 1: Comparison of constitutive equations (adapted from (Tucker, 1989))

Constitutive equations	Small strain response	Viscometric flows η	Viscometric flows ψ_1	Viscometric flows ψ_2	Sudden deformation	Elongational flow	Relaxation after shearing
GNF	useless	very good	useless	useless	useless	useless	useless
CEF	poor	very good	very good	very good	useless	poor	useless
Differential models	moderate	poor	poor	poor	poor	useless	moderate
Integral models	very good	poor	poor	poor	moderate	useless	moderate
Lodge models	very good	poor	poor	poor	moderate	useless	moderate
Linear viscoelasticity	very good	poor	useless	useless	poor	useless	poor
BKZ	very good	good	good	good	very good	good	good

Legend:
GNF: Generalized Newtonian Fluid
CEF: Criminale-Ericksen-Filbey
BKZ: Bernstein-Kearsley-Zapas
η: viscosity
ψ_1: primary normal stress coefficient
ψ_2: secondary normal stress coefficient

While Table 1 clearly contains significant engineering experience and knowledge, nevertheless many questions are left unanswered or informally specified. To begin with, the descriptive terms used in such tables (even with some quantification) have to be *interpreted*, not only in the context of their original compilation, but also in the context of their current use. How, for example, does one decide between using the Integral or the Lodge models, given that their evaluations are the same as shown in Table 1? Are there any exceptions in which a poor model performs well in situations other than those for which it is commonly poor? Are there any tradeoffs involved in the use of these models? For example, are some good models time intensive while moderate models fast so that they can at least be used in preliminary analysis? Are moderate models less sensitive than very good models to the particular problem approached? While the answers to these questions may be found for limited situations by tracing the sources of the table entries in various experimental and theoretical reports, the answers cannot be obvious in selecting an equation to be incorporated, for example, in a finite-element analysis of a particular design. Consequently, designers become aware of the assumptions and attributes of these models through experience, practice, and communication, and hence gradually learn to predict the consequences of using them.

2.2 Comprehensive engineering models

At the other end of the spectrum from models of basic material behavior are models of complete artifacts and design processes. It seems obvious that complications at least as severe as those encountered in the use of basic models will be manifest in the modeling of complete systems. To make things simpler, we will discuss modeling activities in a seemingly "mature" design practice: the design of ship hulls within naval architecture.

In naval architecture, as in few other design practices, models have evolved over many years and have been incorporated gradually into practice. The conservative nature of naval architecture driven by two critical design considerations (cost and risk) led to the careful compilation and saving of significant experience over many years. Nevertheless, as we show through examples from Fritts et al. (1990), there are significant omissions in the compiled knowledge to a much greater extent than those observed in the seemingly basic models of material behaviors.

The design process. Before we start, we must provide a brief overview of the design problem discussed: the design of ship hulls. The task is to define an enclosed shape with the following properties: displace water equal to the total weight of the ship, be stable, have small resistance, behave comfortably in sea waves, be easy to maneuver, and induce small loads on the ship structure. These properties are interlinked. For example, increasing the resistance necessitates installing a larger engine to maintain speed, which in turn, increases the total weight of the ship that must be compensated for by increasing the hull dimensions. Each of these actions have an impact on the stability, seaworthiness, or maneuverability of the ship.

The complexity of ship hull design requires an iterative process in which different levels of approximate MA are used. These MA techniques vary in (1) the *accuracy* of their results, (2) their *applicability* to a particular design phase in terms of concerns such as time spent on preparing the model and executing the analysis, and (3) their degree of validity with respect to the designers' *intent*. These three dimensions roughly determine whether a particular modeling approach is used in specific design phases such as feasibility, preliminary, and contract design. The last evaluation criteria mentioned in Section 1.1, mutual consistency, seems less critical as we show shortly.

Artifact models. The object being designed can be modeled in a variety of ways. The hull can be modeled by its principal dimensions (e.g., length, width, draft, and few other coefficients) (Taggart, 1980); most experimental data exist for this type of model and it is used mainly in feasibility studies of ships. This is possible not because one does not require great accuracy in feasibility studies, but because experience is sufficient to provide accurate predictions with such minimal data as the principal dimensions. The hull can be modeled by offsets: detailed dimensions of hull shape cross-sections at various locations along the hull. Although some experimental data exist for this model type, it mostly serves as an input to quantitative (formal) models. This model is used often in preliminary design. A variation on the offsets model includes the description of the hull appendages such as rudder, fins, and propeller. Finally, the hull can be modeled by a scaled model or by a full size hull; these models are mostly used in the contract design phase.

MA activities. Many MA procedures can use artifact models. These procedures vary in the sophistication of the mathematical formalisms they employ: from simple algebraic relationships imposed over the principal dimensions and critical design objectives to 3-dimensional differential equations used with the offsets description. Model tests are a category by themselves that serve as validation of mathematical MA procedures.

Modeling procedures also differ in the amount and precision of information they generate: from gross values in *empirical* techniques relying on the compilation of significant experience and statistical modeling, through more refined values in *preliminary* techniques, to very detailed and precise values in *detailed* techniques such as sophisticated mathematical modeling and scale model testings.

It is critical to observe that an increased mathematical sophistication of models or the wealth of information they generate does not automatically mean that designers prefer them over, or that they are more accurate than, less sophisticated models. There are several issues that influence their choices, including:

1. *Availability of information.* In early design stages limited information is available, thus, simpler models may be used. In contrast, complex models must *assume* some defaults for the missing information for their execution.

2. *Validity of procedures.* Even if the information is available, it is not clear whether the results of sophisticated (and mostly new) techniques are useful since many of them have been hardly validated. In contrast, simpler models, the result of compiling experience over many years, can provide reliable and validated information for many design decisions. Furthermore, the mathematical sophistication of the models is not an indication of the accuracy of the results because the use of any of the models is subject to many critical informal decisions such as how the sea conditions should be represented in the analysis (e.g., regular vs. irregular waves, combinations of wave directions and heights).

3. *Cost of procedures.* The use of sophisticated techniques is often costly and cumbersome. Moreover, the interpretation of the enormous amount of output some of them generate is subjective.

In summary, the choice between empirical approaches and more formal or sophisticated engineering equation-based approaches, when the choice is available, is governed by other than purely engineering criteria such as satisfying desired function, calculating behavior, or maintaining theoretical rigor, to include criteria of relative cost, time, complexity of use and interpretation, etc.

Fritts et al. (1990) discuss the use of various models and analysis techniques in the design of ship hulls. Table 2 displays a summary of the class of techniques that are used for each of these design phases and each design aspect (i.e., resistance, propulsion, seakeeping, maneuvering, and sea loads).

What does Table 2 tell us?

1. The newer or less common a design is (e.g., advanced monohull, submarine) the less empirical or preliminary procedures are available. This is, of course, true virtually by the definition of empirical approaches.

2. The more traditional the design, the more empirical or preliminary can procedures provide reliable information for making design choices. In the design of conventional hulls, preliminary procedures are sufficient for many design concerns.

3. Different design concerns are amenable to different procedures some of which may not be mutually consistent. This may be the result of the simplicity of a particular design concern or the availability of different experiences.

Table 2: Level of MA techniques vs. design phases (adapted from (Fritts et al, 1990))

Ship type	Conventional monohull					Advanced monohull					Submarine				
Design aspect	R	P	S	M	L	R	P	S	M	L	R	P	S	M	L
Feasibility study	E	E	E	–	–	E	E	P	–	–	E	E	–	–	–
Preliminary design	E	P	P	–	P	D	D	D	P	P	D	D	–	D	D
Contract design	D	D	P	P	P	D	D	D	D	P	D	D	D	D	D

Legend:
Design aspects: R: resistance, P: propulsion, S: seakeeping, M: maneuvering, L: sea loads
Nature of techniques: E: empirical, P: preliminary, D: detailed
– not relevant

4. The preferable trend is to gradually use empirical or preliminary procedures for advanced design stages as exercised in the design of conventional monohulls. Where experience is lacking, detailed, less validated or reliable, procedures are slowly transferred from research to practice. This, however, takes significant time to impact actual design. The time to develop a technique in research, transfer it to practice, and use it in a design project can be 20 years (Fritts et al, 1990), and even then, these techniques maintain the peculiar properties of MA discussed before.

The specification of the type of techniques used in a particular design stage for a particular ship type by no means determines the choice of what model to employ. Table 3 contains details on the specific theories or broad categories of techniques available for each design concern and for each design stage. Note the significant variety of possibilities of such broad categories and the availability of potentially many techniques implementing these broad categories.

Note that the representation of this slow diffusion into contract design as reflected in Tables 2 and 3 is misleading since it conveys an implicit statement that detailed techniques are more accurate than preliminary or empirical techniques. We have already argued against this view. Moreover, in many cases the use of detailed techniques in contract design are reflections of the particular contract, rather than based on engineering necessities.

There is another level of modeling that is manifest in engineering, namely, the modeling of theories in computer codes. Not one of the techniques in Table 3 is performed by hand; rather, engineers use a variety of computer programs that implement a numerical solution of the governing equations of the theories. Again, the underlying assumptions made in codes and procedures are often hidden from practitioners. Therefore, even if one, wrongly, assumes that the theories are precise, their actual solutions in programs are not guaranteed to be

Table 3: Methods of assessment vs. technology areas (adapted from (Fritts et al, 1990))

	Resistance	Propulsion	Seakeeping	Maneuvering	Sea loads
Empirical assessment	Standard series Coefficient algorithms Regression analysis	Standard series Coefficient algorithms Regression analysis	Coefficient algorithms Rank estimators	Coefficient algorithms Regression analysis	Coefficient algorithms
Preliminary assessment	Wave resistance - Thin ship - Slender ship Viscous - Empirical	Lifting line	Strip theory - Small amplitude - 2-D potential flow - Wall sided - Linear free surface - Frequency domain	Coefficient algorithms Force moment balance	Deterministic waves
Detail assessment	Wave resistance - Neumann-Kelvin - Dawson Viscous - Empirical - Model tests	Lifting line Lifting surface Panel methods Model tests	Model tests - Tethered - Radio controlled	Coefficient algorithms Regression analysis	Model tests - Segmented model - Panel cages - Rigid vinyl

accurate. It is clear that computer programs that implement MA procedures must make their assumptions explicit and be tested and validated as other models are.

Altogether, the number of formal techniques available as computer programs and the vast amount of information they produce make their use rely on significant informal information, more difficult, and even risky, contrary to the desires of a conservative profession. Two necessary conditions for appropriate use of formal MA appear to be (Odabasi's discussion of (Fritts et al, 1990), p. 492):

1. Active participation by the end-user community so that correct problems are identified, formulated, and solved;
2. Training of the end-user community — considered one of the most important issues — with sufficient resources being available for this purpose.

The first condition, also adopted in our work (Reich et al, 1992), ensures that pressing problems of design practice are addressed instead of artificial or toy problems. It also ensures that the purpose of MA — pragmatic results — is not forgotten and replaced by work on elegant but unusable tools. Fritts et al. illustrate this pragmatism by discussing the use of coarse-grained models that provide results that differ from reality which are, nevertheless, used successfully by designers. The reason for this success is that the models are sufficient for making comparisons between alternatives, although they fail in producing results accurate in some absolute sense.

The second condition reemphasizes that sophisticated techniques in the absence of proper understanding can result in misapplication. To illustrate this point, Odabasi points to an example in Fritts et al. that displays improper use, as he argues, of a numerical technique. A designer who is unaware of the intricacies of numerical techniques can easily produce erroneous results, both qualitatively and quantitatively.

Our previous discussion centered around techniques in a domain with a long documented history of codes and artifacts. Other domains, such as mechanical engineering, have not evolved such comprehensive understanding of their activities and, as a consequence, the informal knowledge in these domains is even less codified as it were.

3 An Evolutionary Approach to Transforming Theory into Practice

As discussed before and in the next section, the major aspect of MA is the role played by the assumptions implicit in the theories underlying models, the procedures implementing these models, and the designers' activities when they select particular MA techniques, employ them, and interpret the results. In order to take these assumptions into account when developing new means for MA, new ways of developing procedures are required. In particular, the common approach that starts by posing questions such as: "What aspects of preliminary analysis can/cannot be automated?" or "What do engineers require from preliminary analysis computer tools?" are probably inappropriate.

Worse, these questions are premature since they assume that significant research on descriptive design methods has been done (e.g., the question of "what are the types of preliminary analysis used in practice?" has been adequately answered), or they simply ignore practice by holding tight to prescriptive design methods.

In fact, from our perspective, we as engineers *designing tools for supporting MA*, are at the preliminary stage of understanding the problems. If we reflect upon our activities we would acknowledge how little we understand about which MA tools can aid *us* in our design. Therefore, it is premature to attempt to address the requirements of engineers from MA tools. We view the requirements of a design support system as arising mainly from how models, both formal and informal (Piela et al, 1992), are used in context. In light of this, we have adopted a collaborative, evolutionary prototyping approach in order to situate, as much as possible, both ourselves and the tools we create in the working context. As this approach progresses, further research directions that may impact practice are uncovered by designers themselves.

Since we do not attempt to identify designers needs *a priori*, we must approach question of whether MA can or should be automated empirically. This is not to say that automation of elements in the design process is not desirable or achievable; it *is* to say that we *should* collect information on design that will expand our understanding of design as a whole and, as a consequence, enable us and the designers to identify the scope of preliminary analysis. What can and ought to be automated is one question among many others. This applies both to us, as students of design, and to designers themselves.

If we really wish to make an impact on users, our research should take a participatory mode (Reich et al, 1992) — the development of theory being coterminous with participation in and understanding of practice (Floyd et al, 1989; Namioka and Schuler, 1990; Muller et al, 1992). Therefore, we do not view automation as a goal, but as an option, whose specific coverage is usually different from case to case and needs to be guided by context-specific strategy devised by designers to meet the exigencies of particular design situations. In short, automation should not *drive* context — automation should be *driven by* context.

4 Design support requirements: implications of the expanded view of MA

One of the essential properties of modeling is representation. Modeling an entity involves representing it in ways that, hopefully, reveal important properties of the entity being modeled. The behavior of the entity may be calculated through the use of simulation tools formally, or the past performance of entities similar to the one at hand can be discovered informally.

Since designers use a variety of MA procedures, it is our contention that no single representation or abstraction technique can be imposed on designers *a priori*, without severely limiting their ability to model. We thus use a notion of conceptual information modeling that allows multiple classifications to be imposed over a corpus of information. Abstraction levels are imposed by the users, in whatever way they see fit.

Since designers use a variety of representations to model and analyze designs depending on the types of functionalities required in the performance of the task, we have built *n*-dim to support the incorporation of any tools designers find appropriate to carry out the above activities. This allows *n*-dim to benefit from research on numerical MA developed within engineering disciplines as well as from research on symbolic MA, such as qualitative physics, developed in AI. *n*-dim is being developed to facilitate modeling starting from the initiation of a design process and continuing throughout the life-cycle of the artifact (Levy et al, 1993; Subrahmanian et al, 1991).[4] Supporting this integration capability and insisting that *n*-dim

[4]The third generation of *n*-dim is currently built in a participatory evolutionary prototyping mode: we encourage users to use the tool and participate in its development; we use it to model and implement its design in several ways, including issue-based models (like gIBIS, Conklin and Begeman (1988)), models of the actual implementation of the software (decompositions in terms of class hierarchies, functional requirements, documents, etc.) and other kinds of information; and we introduce changes incrementally, rather than

maintains its usability and scaleability requires addressing significant problems in diverse areas such as visual programming, distributed databases, graph grammars, human-computer interaction, and machine learning.

So far, given these objectives, the development of *n*-dim has de-emphasized artificial intelligence (AI). We are, however, using as elements of our work, techniques from AI such as semantic network representations, rule structures, machine learning techniques, and other techniques and representations. Techniques such as relational databases, hypermedia, graph grammars, etc. can be used to empower the user to organize, conceptualize, and reason over (including model) information.[5]

While design is a social process, it also takes place in a larger social context. Thus, two types of hurdles need to be overcome in applying our technique to real life problems: organizational and technical. Our contention is that the organizational is more important than the technical seemingly sound techniques fail constantly in practice due to lack of attention to organizational issues. Our development approach—participatory design and evolutionary prototyping—is geared towards alleviating this problem (Reich et al, 1992), while the techniques implemented in *n*-dim are meant to give designers the ability to model and analyze their organization, the interactions with their peers, and the flow of information within the organization.

5 A description of *n*-dim

In this section, we attempt to give first a brief overview of the way in which *n*-dim allows one to model information, and then elaborate on certain key elements of these modeling facilities, as well as operational issues associated with using them. For a more detailed discussion of the implementation of the system (Levy et al, 1993).

There are many analogies that can be drawn between *n*-dim and other types of systems. One can think of *n*-dim as providing:

- A hypertext-like system with typed, first-class links;
- A large, extensible, distributed rule-based system with versioning capabilities;
- A configuration management and revision control system built on top of a relational database.

abruptly.

[5]One would not be wrong if one detects here a statement against “strong” AI, with, however, an appreciation of AI tools.

The following will focus on the three main aspects of *n*-dim that, when combined, seem to form a of critical mass which together imbue *n*-dim with its special enabling character: a *flat space* of objects, a *generalized* notion of modeling that extends from prototypes to classes in a uniform way, and the semantics of *publication*.

5.1 Flat Space

The space of objects in *n*-dim is conceptually flat; that is, objects do not contain other objects, *per se*. Instead, multiple structures can be imposed on this flat space by means of *models*, which are comprised of *links*, or relationships between objects (models themselves being objects). In this way, the same object may participate in many models.[6] Since *n*-dim models are nothing but linked information objects, they enable the capture of the rich and complex formal and informal contexts of a given object and hence, *n*-dim models can be the bridges between formal and informal modeling as discussed in the previous sections.

There is a basic cleavage in the space of *n*-dim objects between *atomic* and *structured* objects. As the name indicates, *atomic* objects cannot be broken down any further, e.g. an integer, a link, a piece of text, an image, an audio bitstream, etc. One could think of atomic objects as things that have *values* of some sort.[7]

The primary form of structured object is the model. A model is a set of links, which are, themselves, atomic objects. The value of a link object is a 3-tuple, [*source,target,type*], where *type* is merely a label for the link; link types are given their meaning(s) by the modeling language(s) in which they occur.[8] It is quite possible to have the same link type mean totally different things in different contexts; we view the meaning of links as something to be *negotiated* by users of the system over time. Operationalizing the semantics of particular interpretations of links is considered an open-ended process; *n*-dim provides mechanisms for doing so, but does not require it to be done in order to use a link type. There is one special link type which is known to the system: the `part` link. By convention, *n*-dim models are rendered in a boxes-and-arrows presentation; `part` links are displayed as boxes inside of boxes, whereas all other kinds of links are displayed as directed arrows. If two models are

[6] *n*-dim is implemented in a prototype-based object system called BOS, the Basic Object System (Levy and Dutoit, 1993). Since it is prototype-based, there are no classes, per se; rather, any object is a potential prototype for another object. For more information on prototype-based object systems, see (Ungar and Smith, 1991).

[7] The creation of new atomic object types generally requires some programming, since new types of values often indicate new types of fundamental operations. The suite of built-in atomic object types, while not completely exhaustive, is rich enough to make building up new kinds of objects from the existing set at least possible.

[8] Section 5.2, below

parts of a third model, the parts of the two included models are not visible from the third model. The parts of the included models cannot be linked in the including model; only objects which are explicitly represented as parts of a model can be the source or target of links.

5.2 Roles of a Model

Models play (at least) two roles in *n*-dim: instance/prototype and language.[9] In its role as a prototype, a model can be conceived of as an object in a prototype-based object-oriented system such as SELF (Ungar and Smith, 1991); prototypes may be *copied* as starting points for new models. In addition, every model can be viewed as representing a *class* of models in a generative sense; that is, the set of links and objects used in the model become the vocabulary, and the (embedded) rules of composition become the syntax and scope of semantics for building other models. In this sense, a model serves as a language. All objects refer to another model as their *modeling language*, and are said to be *in* that language. The only kinds of objects and links that can be put in a model are those mentioned in its language, and only legal compositions of these objects and links can be created.[10] A model viewed as a modeling language can be thought of as a grammar. More formally, this grammar defines:

- The set of legal parts which models in that language may contain;
- The set of legal link types or labels between parts of models in that language;
- Rules of composition (model-building) for the set of objects and the set of links.

In *n*-dim; any object can be used as a modeling language. If, for instance, one were to ask *n*-dim to use an `Integer` object[11] with the value 1 as a modeling language, one would get an object in the language `1`, which could only have as its value the number 1. This grammar has only one legal sentence, which happens to be the grammar itself.

5.3 The *n*-dim notion of *published*

When a user of *n*-dim creates an object, it is theirs, and theirs alone. No other user of the system can even know of its existence until its originator *publishes* it. Publishing an object

[9] We will use the terms "instance" and "prototype" somewhat interchangeably in what follows, since, in a prototype-based system, the two concepts coincide. However, the different connotations are useful in distinguishing various *uses* of the word "model".

[10] However, the built-in `Universal` modeling language provides a way around these restrictions (see below).

[11] Note that `Integer` objects are atomic!

makes it simultaneously *immutable* and *visible*[12] to the rest of the user community. In order for a model to be published, the targets of all `part` links with it as their source must also be published[13]. In order for a model to be used as a language, it must first be published.

Publishing an object in *n*-`dim` has precisely the same effect as publishing a paper: it becomes a part of the collective and ceases to be the sole property of its originator. Just as it is not possible to remove from the libraries of the world all of the copies (and copies of copies) of a paper once published, so it is not possible to alter the status of an object in *n*-`dim` once published. If a revision (or retraction!) is needed, then the published object must first be *copied*, and the copy revised and subsequently published. Whenever any object is copied, *n*-`dim` automatically creates or updates a *pedigree* model (constructed in a built-in language called `Pedigree`), which contains `part` links to all relatives of the object and `copy-of` links between them, which carry the trace of its evolution through time. A copy of an object can evolve and change so drastically that it no longer resembles its immediate ancestor.[14]

The fact that models must be published to be used as languages is significant. One of the key notions is that when a user goes back to look at an object published a year ago, it should behave *as it did a year ago*. This becomes extremely important when one relates the volume of information that designers and engineers must typically deal with to the length of time being considered. At any given time, most people can deal with a few pages of text or a figure or two at a time, contrasted with the immense amounts of information that a single person can generate and work with over the course of a month or a year. In this sense, collaborating and negotiating with *oneself* can become a major issue. Since models must be published in order to be used as languages, one is guaranteed that any model, once published, will never have its corresponding modeling language changed "from underneath" it.[15] Also, modeling languages being designed artifacts in the system, like any other object they too have their history captured via publication.

[12]There is a system of access controls available, a full treatment of which is beyond the scope of this document. It should be noted, however, that access control is, like nearly everything else, done declaratively through the creation, publication and revision of *n*-`dim` models

[13]There is a recursive variant of the publication operation to make this slightly less onerous.

[14]Of course, the modeling language used to construct the original also apply to copies, so this is not as radical a notion as it may appear to be.

[15]If one thinks of a modeling language as being akin to a compiler for a traditional computer language, and models in that language as programs, then the essence of this guarantee is that when a program written years ago is used, the version of the compiler current *at the time the program was written* will be in effect.

5.4 Other Aspects of the Representation

Although space is too limited to present a complete description of *n*-dim, a few other points are worth noting.

Structure, Projection and Presentation. First, all models actually have a tripartite representation, referred to as the *structure*, *projection* and *presentation* layers. The structure of a model is simply its links. One can map a structure of an *n*-dim model onto multiple *projections*, which discriminate between possible views of that structure. Any projection can be mapped onto multiple *presentations*, which fix the characteristics of a projection *vis a vis* its rendering. A rendering of a model can be something like a window presented to the user for interaction, a printed file, etc. Projections are models, as are renderings. *n*-dim merely *interprets* these models appropriately when needed.

The `Universal` *modeling language.* There is a special modeling language built-in to *n*-dim called `Universal`, which is special in the sense that models constructed using it as their language can contain any kind of object or link. There are two main uses for `Universal` models: creation of "folder"-type models, used merely to organize other objects, and creation of models meant to be used as modeling languages. By convention, all modeling languages are constructed in `Universal`, although this is not a requirement.

The `Rule` *modeling language.* The `Rule` language is another in the set of built-in *n*-dim modeling languages. It provides a simple if/then predicate structure for representing in a declarative fashion any of several classes of information, including actions to be taken when a certain event (or pattern of events) is seen by the system,[16] limitations on the construction of models that cannot be easily (or at all) captured by the modeling-language mechanism,[17] etc.

5.5 Status of *n*-dim Implementation and Deployment

It has been our intention from the outset to produce a tool that could be demonstrated to be something more than an academic "toy" by real, industrial users. As of this writing, *n*-dim has seen three major prototype implementations,[18] and is undergoing a fourth. In each generation, one of the major factors driving the design and implementation has been *scale*. The first Unix implementation (version 0.8, 1991) was capable of dealing with hundreds of

[16] User-supplied operations on objects (also called *methods*) can be implemented in this fashion, with the triggering event being an explicit user action.

[17] Such as quantifiers, algebraic constraints, etc.

[18] One as a HyperCard stack on the Macintosh, and two generations of Unix^{TM}-based implementations.

objects and was single-user. The current implementation (version 0.9, 1992-3) currently has tens of thousands of objects stored in it and can support several simultaneous users. The next prototype (version 1.0, due 1993) will scale to hundreds of thousands or millions of objects and will support hundreds of simultaneous users.

The version 0.9 prototype is being used and tested internally by the group and is being deployed both internally at the Engineering Design Research Center at large, and at industrial sites for exploratory use by some of our sponsors. Members of the group have used this version to experiment with integrating external computational agents (tools) into the *n*-dim environment, including:

- A text-based information retrieval system. This is a suite of public-domain programs, some of which were developed by members of the group, which use Natural Language Processing (NLP) techniques to analyze large corpora of text. The suite includes a tagger, which tags every word with respect to its part of speech, a regular-expression-based parser which can extract noun- and verb-phrases from tagged text, term-cluster programs and the beginnings of a semi-automated concept-network-building assistant which can be used from inside of *n*-dim to create concept structures (as *n*-dim models) for specific domains, using the results of term-clustering. In our system, we use the NLP tools in conjunction with the SMART information retrieval engine from Cornell University (Salton, 1971; Buckley, 1985). A graphical interface user interface to the NLP system has been built and integrated into the *n*-dim environment.
- A blackboard-based system for maintaining consistency between parameters used by analysis tools. This system has been used extensively by the group in working with industrial sponsors to build an *n*-dim-based infrastructure for design support which integrates access to existing analysis tools with access to persistent shared memory (e.g. the published object base) (Finger et al, 1993).
- A generative layout system based on hierarchical decomposition called ABLOOS (Coyne, 1991), developed at the Engineering Design Research Center. A group of graduate students spent a semester developing *n*-dim modeling languages that could be used to describe an ABLOOS input problem, as well as represent libraries of standard parts. Methods defined in these modeling languages could automatically produce the LISP source file needed by ABLOOS to run. The user interface to ABLOOS was not altered, but simply invoked from inside of *n*-dim.

In the fall of 1993, version 1.0 is scheduled to be used in support of a senior-level software engineering course which features a team-based approach driven by the use-case methodology developed by Jacobson (Jacobson et al, 1992). Various other collaborative efforts involving the deployment of this and subsequent implementations of *n*-dim are on the drawing boards.

6 MA with *n*-dim: An example

We will now attempt to paint a picture of the use of *n*-dim in a design and manufacturing organization.

We emphasize the informal aspect of modeling, whether they underlie formal methods or not, rather than the use of formal techniques. The incorporation of the latter in *n*-dim can follow procedures similar to those used for incorporating the three external computational agent described in the previous section. The scenario illustrated deals with designing a hypothetical product: a computer that can be carried by an operator along the Alaska pipeline to gather information about the conditions of the pipe.

The design specification document created after discussions with the client provides the engineer with a basis for the initial modeling (see Figure 1; also, a summary of some modeling activities appears in Table 4). The model `customer specs` is constructed to represent some requirements and their relationships. The engineer interprets these requirements as the essence of the design specification document. By extension, the engineer thinks that this model is a good model of the intended design..

The engineer uses this rough model to query the corporate database (i.e., corporate memory) by highlighting some essential objects. The default query mechanism used by the engineer (the model `query execution`) extends the query through an active agent called `augment synonyms` when executing the search; this is done through the use of the aforementioned NLP tools and the `Execution` modeling language which allows the incorporation and managing of external programs. Some synonyms include: `portable` for `carried by hand` or `collect information` for `gather information`. The query is an abstract model of the intended design; its analysis depends on the way the designs retrieved succeeded and its applicability on whether the query can lead to a successful new design. As seen in the model `previous designs`, the search retrieves several designs. Note that only one of them (`car computer`) would have been retrieved had the NLP tools and synonyms not been used to elaborate the query.

The previous designs are studied to see whether any of them has functional requirements similar to those of the present design (thereby allowing their use as prototypes). Through a simple inspection and copying of relevant parts of previously published design issues models, the engineer constructs a `design issues` model for the present design. Later, the engineer

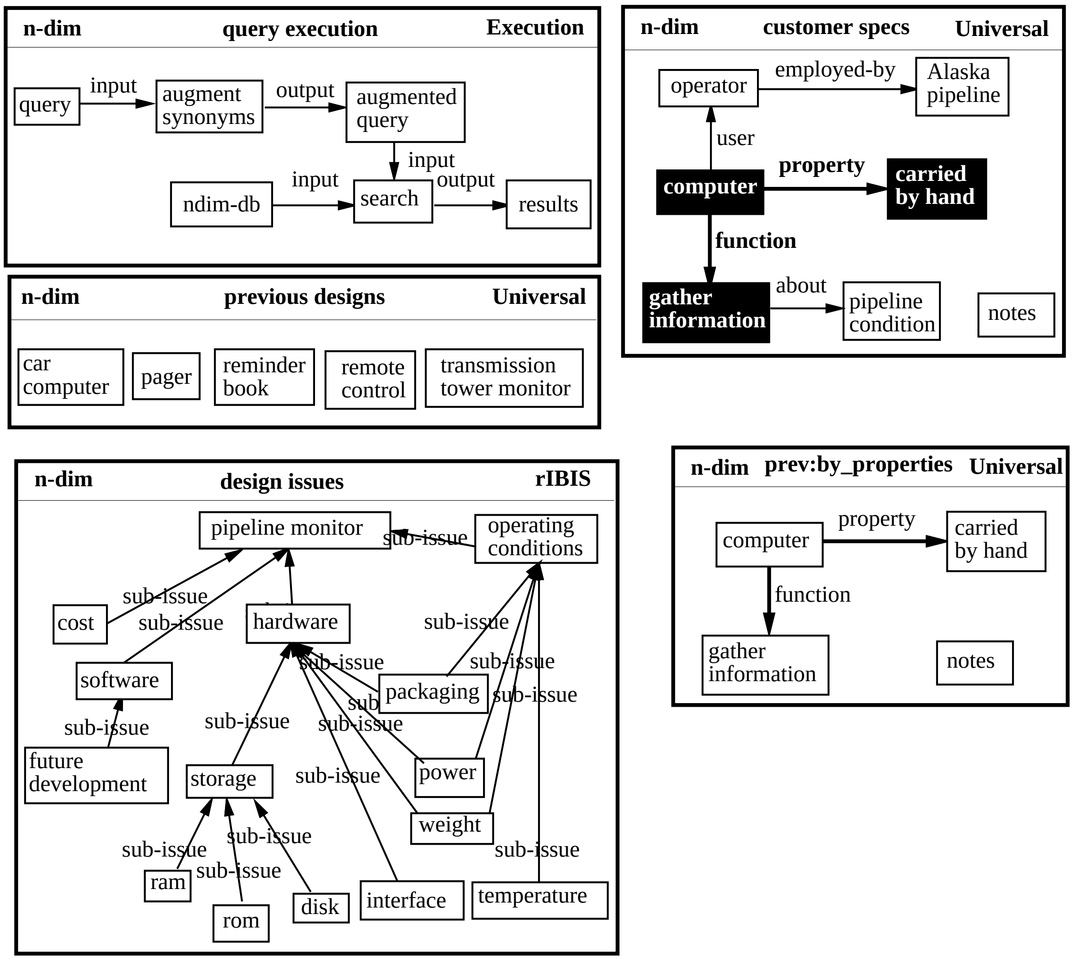

Figure 1: Initial modeling activities

will transform this model into a modeling language for specifying issues for a portable computerized device (thereby using the model as a language). Similarly, the engineer decides that the query was useful and saves it in a separate model (`prev:by_properties`) with some annotations.

Browsing through the design specification document and design issues model of previous designs, the engineer decides that the previous design most similar to the present design is the `transmission tower monitor (TTM)`. The engineer thinks that its design is a good model of the intended design and considers borrowing as much as possible from its functional as well as component design. The `TTM` design is inspected to uncover the critical constraints governing its design. This follows from the assumption that the `TTM` past analyses and constraints checking are useful models of analyses of the present design.

The engineer looks at the models written in the `spec_margin` modeling language shown in

Figure 2. Note that the presentation of the margins is in the form of tables connected by `part` links which simulate hypertext-like links; other presentations such as graphical bars are similarly possible.

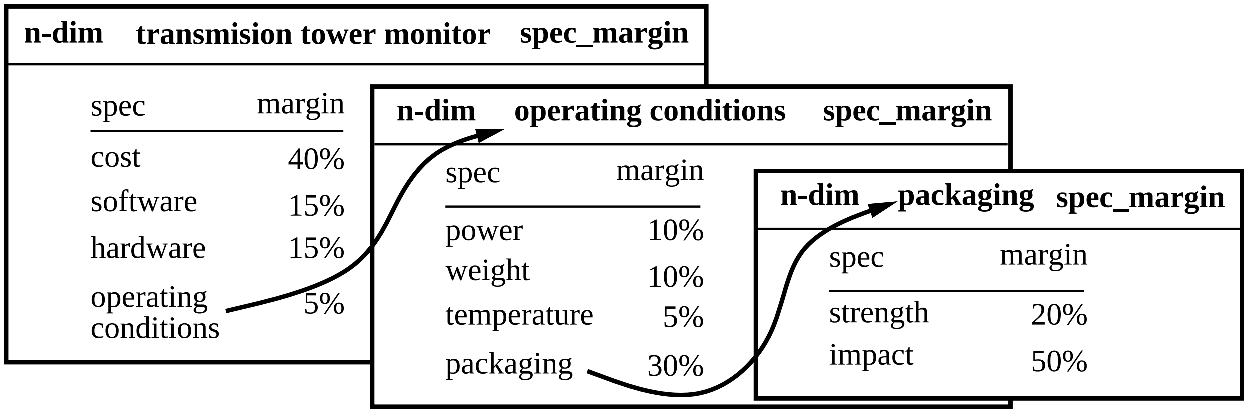

Figure 2: Retrieved margins of safety of an old design

The engineer observes that the `TTM` design was governed by thermal constraints. Further, by tracing the constraint back to the design, the engineer see that ventilation holes were introduced in response to analysis results detecting an overheating of some electronic components near the power supply. The engineer expects that the thermal behavior of the new design will be better due to its lower operating temperature, allowing significantly more heat to dissipate. The functional specifications can be met with minor changes to the old design.

After making the essential changes, the only additional analysis required is mechanical. Since this is outside the expertise of the engineer, the engineer decides to consult with the company packaging expert. The discussion proceeds through the use of model creation, publication, and modification (see Figure 3).

The engineer's `governing issues` model contains the critical constraints and the object `notes` contains the question regarding the status of the strength analysis in light of the low operating temperature. After publishing the model, the engineer notifies the expert of its release. In response, the expert reviews the question contained in `notes` and models the influence between the temperature and the stress safety margin in the `Influence` language that allows engineers to make inferences about influences as well. To better ground the first influence, the expert adds a model of the quantitative relationships between temperature and tensile strength of plastic materials. This model can be merely a graph of the data or a graphic representation of equational quantitative relationships that can be imported into analysis procedures when needed. As discussed in the case studies in Section 2, this graph is also a model because it provides the material properties calculated through experiments

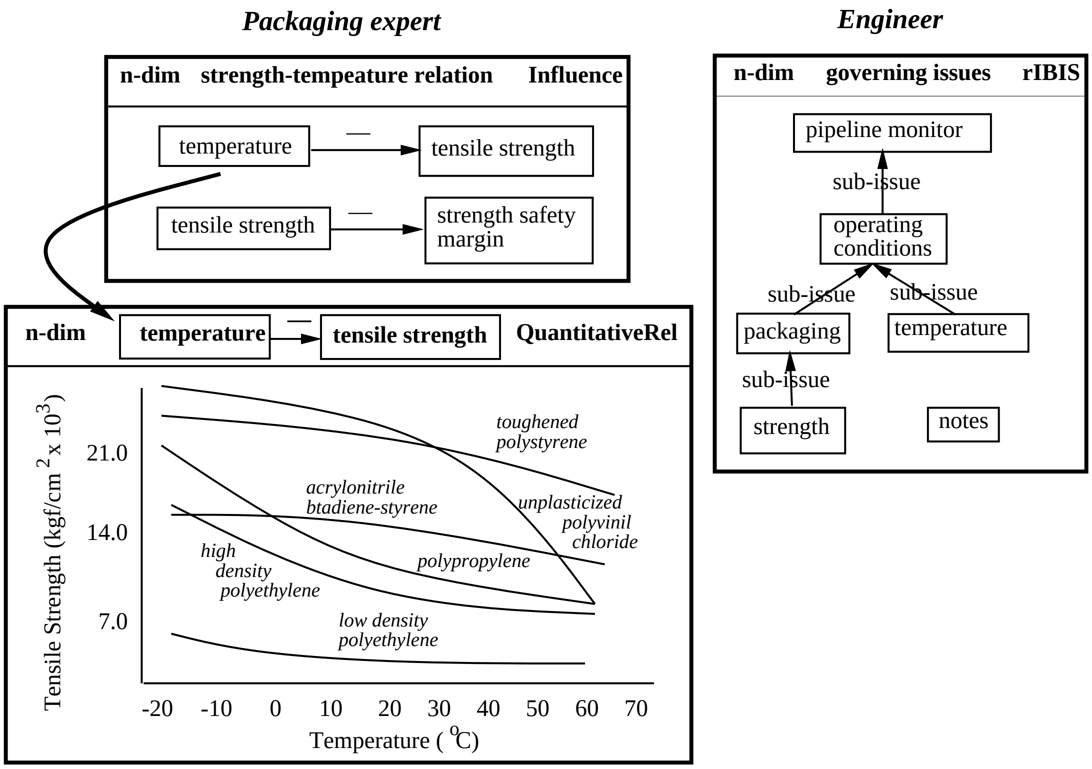

Figure 3: A discussion between the engineer and the packaging expert

conducted on standard specimens under certain conditions. As such they model more diverse operating conditions of materials.

Based on the expert's response, the engineer concludes that the casing of the old design is suitable for the new one. Subsequent to the detailed design, a prototype is manufactured and subjected to laboratory testing specified in the design specification document. These testing procedures model the operating conditions expected to govern the behavior of the device in the field.

During the impact test, the prototype fails by breaking near an air ventilation hole in the vicinity of the power supply base. Clearly, at least one of the models, whether formal or informal, employed failed. It is critical to locate and modify it to avoid recurring failures.

The failure analysis and review detect the following. At a temperature close to -30°C, the impact strength can decrease to less than 30% of its value at a reference temperature of 20°C. Not accounting for this fact marks a failure of `governing issues` created by the engineer to model the complete influence of the temperature on the mechanical behavior of the design. It is also a failure of the expert's model which relied too heavily on the engineer's query without extending the influence of temperature on strength to its influence on impact.

A close examination finds that while the old design passed the testing procedures, few of the products failed in field use due to similar breaks. These few failures despite formal thermal and mechanical analyses followed by successful laboratory testings of prototypes can be traced to failures in the informal underpinning of formal modeling. The specifics of this instance are described below.

When the previous design was analyzed, the engineers did not take into account the stresses due to assembly operations. They only included the stresses due to impact loading in their calculations. The successful laboratory testings and the actual measurements taken were perceived as verifying the results of the analysis. Unfortunately, the laboratory testing procedure was flawed; it did not model the operating conditions well. In the testing, part of the device near the power supply warms up to 50°C. To test durability, this temperature was maintained for a relatively long time period, in which the material gradually undergoes creep through which assembly stresses are relieved. This process presented lower impact stress levels in the subsequent impact test. Clearly, a portable computer can hardly be used for an extended period of time to allow such behavior to occur in the field. Further, an impact load can occur even before operating the computer.

Following this analysis, both designs undergo revisions. In the present design, the ventilation holes are eliminated and in the TTM, a small design modification removes the assembly stresses. Both revisions and their rationales are modeled and published for future reference. Additional links are created between strength, impact, and temperature in the analyses models when designing with polymers.

In addition to the design revisions, the company's engineering division management revised the design process model. It was now mandatory for any expert to provide a comprehensive response to limited queries within their expertise and subsequently review the design. This was done through revising the design process model of the company.

n-dim is being developed to enable the capture, retention, and subsequent access of cases of both formal and informal modeling in design. Table 4 summarizes some of the critical modeling activities discussed in this scenario and their properties. It serves to ground these activities in the discussion in previous sections. The first column specifies the model; the second specifies the nature of the model (i.e., formal or informal) and its representation (i.e., structured or unstructured); the third specifies the object being modeled; the fourth how the model is analyzed and the trigger for its modification; and the fifth specifies how the applicability of the model is determined.

The character of the engineering design work as illustrated in this scenario exemplifies a number of MA activities, formal and informal, that interact with each other. The interaction and organization of these activities in engineering and design tasks is not determinable *a*

Table 4: A summary of MA activities

Model	Type	Object modeled	Analysis and refinement of model	Applicability
design specification document	informal/ unstructured	intended design	discussions between customer and engineers	success of product
customer specification	informal/ structured	design specification document	review of design documents by customer	customer approval
query	formal (underlying informal)/ structured	intended design	success of the designs retrieved	success of intended design
analysis results of old design	analysis — formal (underlying informal), interpretation — informal/structured	analysis of intended design	usefulness for designing new design	passing testing procedures
discussion engineer expert about temperature stress relationship	informal question/ structured and unstructured, formal presentation of response (underlying informal)/ structured	stress behavior of intended design	testing procedures	success of product
testing procedures	semi-formal (underlying informal)/ unstructured	environmental operating conditions	measurements in field operations	survival of products
design process model	semi-formal/ structured	optimal design process	internal review	failures of designs due to organizational issues

priori. In this paper, we have described an environment that allows for the capture of these interactions in as ubiquitous a manner as possible. Furthermore, the activities described and their codification while they are used in design provide rich data for studying MA activities in design and for further improving n-dim.

7 Discussion and Conclusions

We view the following as central research goals and challenging problems:

1. What are the types of models and analyses used by designers?
2. What types of representations (modeling) and techniques are required to perform such analyses?
3. How can the necessary data be collected to address (1) and (2)?

4. How does one create tools, automated or not, which give designers the ability to improve their own practice?

It is our hope that this paper has outlined an approach for addressing all of the above.

We have described an approach to the study of the practice of designers, and, further, to the development of tools that aid designers throughout the design process. This approach will make use of whatever MA tools designers wish to use, integrating available techniques or, possibly, formulating new problems when available techniques prove inadequate. We have argued that the majority and most critical of MA activities in design are informal, and, further, that even in the course of using supposedly formal techniques, a wealth of less formalized, codified or even understood knowledge must be brought to bear in order to successfully apply, for instance, equational modeling techniques. Therefore, we have focused our work on developing support tools for informal MA, including a generic information structuring tool called *n*-dim, which serves as the medium through which information, tools and, most importantly, people interact. We have throughout our work applied the maxim *whatever works* [19] as a means of grounding ourselves in praxis. In this light, we view AI, NLP, machine learning, information retrieval techniques, and all the other accumulated techniques, practices, and tricks of the trade as being interesting in so far as they are *useful* to the twofold task at hand, namely, studying design, and attempting to help designers help themselves. Of course, the almost infernally rich, varied and chaotic wealth of information and experience available from the engineering disciplines themselves, especially in applied settings, provides us with constant inspiration and problems.

It is clear that considering either formal or informal techniques in isolation from practice is insufficient to address the decision support problem. We advocate and follow a methodology for developing design support systems: the participatory design approach. We have developed and are continuing to develop *n*-dim as a tool for actual use in several engineering domains.

Our participatory design approach in the design of *n*-dim itself provides the answer to the issues raised at the beginning of this section. Since *n*-dim is meant to be used in actual engineering practice, it will serve as a repository of techniques that are used by designers. The representations and techniques to support these activities will be developed in an evolutionary manner, and the trace through time of their development will also be captured. *n*-dim will also serve as a testbed for studying the activities of designers in the course of their work. Our belief is that the approach we take in the development of *n*-dim will result in a successful system that supports design, including the MA activities. As such it may serve as a model for the development of support systems for other tasks as well.

[19] Or, in more polemical moods: *anything goes!* (Feyerabend, 1975).

Acknowledgments

E. Subrahmanian, S. N. Levy, and A. W. Westerberg are supported by the Engineering Design Research Center, a National Science Foundation Engineering Research Center. The views and conclusions contained in this document are solely those of the author(s) and should not be interpreted as official policy, either implied or expressed, of the SEI, CMU, the US Air Force, the Department of Defense, or the US Government.

References

Bucciarelli, L. L. (1988). "An ethnographic perspective on engineering design." *Design Studies*, 9(3):159–168.

Buckley, C. (1985). "Implementation of the smart information retrieval system." Technical Report 85-686, Cornell University, Department of Computer Science.

Conklin, J. and Begeman, M. L. (1988). "gIBIS: A hypertext tool for exploratory policy discussion." *ACM Transaction on Office Information Systems*, 6(4):303–331.

Coyne, R. F. (1991). "ABLOOS: A computational design framework for layout synthesis." PhD thesis, Department of Architecture, Carnegie Mellon University, Pittsburgh, PA.

DeMillo, R. A., Lipton, R. J., and Perlis, A. J. (1979). "Social processes and proofs of theorems and programs." *Communication of the ACM*, 22:271–280.

Feyerabend, P. K. (1975). *Against Method*, New Left Books, London, UK.

Finger, S., Subrahmanian, E., and Gardner, E. (1993). "A case study in concurrent engineering for transformer design." In Rosenburg, N. F. M., editor, *Proceedings of ICED-93 (The Haugue)*, pages 1433–1440, Zürich, Heurista.

Floyd, C., Mehl, W.-M., Reisin, F.-M., Schmidt, G., and Wolf, G. (1989). "Out of Scandinavia: Alternative approaches to software design and system development." *Human Computer interaction*, 4(4):253–350.

Fritts, M., Comstock, E., Lin, W.-C., and Salvasen, N. (1990). "Hydro-numeric design: Performance prediction and impact on hull design." *Transactions SNAME*, 98:473–493.

Hales, S. (1987). "Analysis of the engineering design process in an industrial context." PhD thesis, Department of Engineering, University of Cambridge, Cambridge, UK.

Jacobson, I., Christerson, M., Jonsson, P., and Övergaard, G. (1992). *Object-oriented Software Engineering – A Use Case Driven Approach*, Addison Wesley, Reading, MA.

Kanigel, R. (1991). *The Man Who Knew Infinity*, Charles Scribners and Sons, New York, NY.

Konda, S., Monarch, I., Sargent, P., and Subrahmanian, E. (1992). "Shared memory in design: A unifying theme for research and practice." *Research in Engineering Design*, 4(1):23–42.

Leifer, L. J. (1991). "Instrumenting the design process." In *Proceedings of ICED-91 (Zurich)*.

Levy, S. and Dutoit, A. (1993). "BOS: the basic object system." In progress.

Levy, S., Subrahmanian, E., Konda, S. L., Coyne, R. F., Westerberg, A. W., and Reich, Y. (1993). "An overview of the *n*-dim environment." Technical Report EDRC-05-65-93, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Muller, M. J., Kuhn, S., and Meskill, J. A., editors (1992). *PDC'92: Proceedings of The Participatory Design Conference (Cambridge, MA)*, Computer Professionals For Social Responsibility, Palo Alto, CA.

Namioka, A. and Schuler, D., editors (1990). *PDC'90, Proceedings of The 1990 Participatory Design Conference (Seattle, WA)*, Computer Professionals For Social Responsibility, Palo Alto, CA.

Petroski, H. (1992). *To Engineer is Human*, Vintage Books, New York. (First Edition, 1982).

Piela, P., Katzenberg, B., and Mckelvey, R. (1992). "Integrating the user into research in engineering design systems." *Research in Engineering Design*, 3(4):211–221.

Reich, Y., Konda, S., Monarch, I., and Subrahmanian, E. (1992). "Participation and design: An extended view." In Muller, M. J., Kuhn, S., and Meskill, J. A., editors, *PDC'92: Proceedings of the Participatory Design Conference (Cambridge, MA)*, pages 63–71, Palo Alto, CA, Computer Professionals for Social Responsibility.

Salton, G. (1971). *The SMART Retrieval System — Experiments in Automatic Document Processing*, Prentice-Hall, Englewood Cliffs, New Jersey.

Sargent, P., Subrahmanian, E., Downs, M., Greene, R., and Rishel, D. (1992). "Materials' information and conceptual data modeling." In Barry, T. I. and Reynard, K. W., editors, *Computerization and Networking of Materials Databases: Third Volume, ASTM STP 1140*, American Society For Testing and Materials.

Subrahmanian, E., Westerberg, A. W., and Podnar, G. (1991). "Towards a shared information environment for engineering design." In Sriram, D., Logcher, R., and Hukuda, S., editors, *Computer-Aided Cooperative Product Development, MIT-JSME Workshop (Nov., 1989)*, Berlin, Springer-Verlag.

Taggart, R., editor (1980). *Ship Design and Construction*, The Society of Naval Architects and Marine Engineers, New York, NY.

Tucker, C. L., editor (1989). *Fundamentals of Computer Modeling For Polymer Processing*, Hanser Publishers, Munich.

Ungar, D. and Smith, R. B. (1991). "SELF: the power of simplicity." *LISP and Symbolic Computation*, 4(3):187–205.