Computational Quality Function Deployment *is* Knowledge Intensive Engineering

Yoram Reich
Department of Solid Mechanics, Materials, and Structures, Faculty of Engineering, Tel Aviv University,
Ramat Aviv 69978, Israel, Phone: +1 972 3 640 7385, Fax: +1 972 3 640 7617, E-mail: yoram@eng.tau.ac.il

To appear in *The Proceedings of KIC-1 (Knowledge Intensive CAD)*, IFIP WG 5.2, Helsinki, September, 1995.

Abstract
This paper describes the development of computational support tools for practically successful engineering techniques. The paper reviews the requirements for manual Quality Function Deployment techniques, presents them, and discusses their limitations. It argues that computational support tools can alleviate most of these limitations and that a graph-based information representation for such techniques is an excellent choice for supporting both QFD techniques and their integration with other external CAD-related computational services. The paper presents an architecture for a computational QFD (CQFD) tool based on the graph-based modeling environment *n*-dim. It shows how this architecture supports most of the requirements for QFD techniques, in addition to providing many additional functionalities, and briefly illustrates how the CQFD tool will be used.

Keywords
Quality Function Deployment, TQM, design practice, 7 management tools, graph representation, *n*-dim, collaborative design, design rational

1 INTRODUCTION

Over the last years there is a constant and growing flux of studies developing intelligent computational tools for supporting or even automating engineering tasks. The underlying belief that new computational tools will provide a competitive edge is so entrenched that in a recent study comparing the CAD practices of American and Japanese engineers, the working hypothesis was that one explanation of Japanese economical success compared to American industries (and beside management style superiority) would be that Japanese

CAD practices are more "advanced" than American practices (Liker et al, 1992). The results of this study did not support this hypothesis. Japanese engineers had less access to CAD tools, but they used them differently: Experienced Japanese engineers made use of advanced CAD features such as 3D modeling as opposed to drafting, whereas experienced American engineers hardly ever used CAD tools. Thus the difference in practice was not due to the availability of better CAD tools but due to different attitudes towards tools that led to their effective use.

Similar observations have been made about other industries or technologies (Ealey, 1994). During a visit to Japan in 1984, a journalist found 20-years-old American-made equipment in one of the most efficient Toyota's car engine plants. When he asked about it, the answer was that when the plant was built, this machinery was the best available. Years later, this equipment worked more efficiently and produced better quality products than comparable American plants equipped with recent high technology (Ealey, 1994).

It seems that for Toyota's engineers, machines start their life performing poorly and thereafter improve with the experience engineers gain with them; technology becomes "mature" as it becomes older and its users become familiar with its "behavior". For American executives, machines performance start degrading immediately after they are manufactured. They will invest in new technology without thinking about the overhead of buying and assimilating it into their organizations. Often, this strategy will result in poor quality performance (Deming, 1993).

These evidences dispute a more general hypothesis that high-end technology is necessary for competitive advantage. In contrast, often, high-end technology is used as a pill: An initial hype about the technology is followed by quick use that may have some immediate benefits or lucky side-effects. Unfortunately, these consequences may disappear quickly (Ealey, 1994). The reason behind competitive advantage is clearly not high-end versus low-end technology, which anyway transfers easily, but the *attitude* towards or the "relationships" with the technology.

The short relationships between CAD research and industry is scattered with techniques that did not mature to be practical or that were not adopted by practitioners. There are several exceptional projects that succeeded due to special circumstances. From them we may conjecture that when technology is set to replace laborious uninteresting manual procedures, whose cost can be clearly identified, it has a high chance to succeed. Also, if the behavior of a technique is obvious or could be explained easily is has a higher chance to be accepted (Fenves et al, 1994; Reich, 1994). Two successful examples are:

1. the move from the manual drafting board to 2D computerized drafting and recently to 3D modeling; and
2. the replacement of manual strength calculations of structures by finite element analysis that was further adapted to solving partial differential equations in diverse fields.

There are other examples where new CAD techniques demonstrated practical results (Umeda et al, 1991; Tomiyama, 1994), or where significant projects, notably the Boeing 777, were designed with computational tools that proved highly valuable; nevertheless, the majority of computational techniques did not make it into the mainstream of engineering use.

In general, approaches to supporting Knowledge Intensive CAD (KIC) can be divided into two complementary paths. The first path explores high-end techniques. Even with an ultimate goal of testing them in practice, based on the previous discussion, these techniques involve high-risk with respect to their practical utility. The second path involves relatively low-end technology and consists of selecting successful manual techniques and improving them through computational support in an evolutionary manner that constantly maintains their practical relevance, usability, and effectiveness. The promise of this approach follows directly from the introduction.

This paper presents an example from the second path: the development of computational support tools for Total Quality Management (TQM). TQM studies seek to develop and experiment with techniques that integrate customer, design, manufacturing, and other life-cycle concerns early on into the design process for designing and producing better quality products that can compete in a world economy market. The design, manufacturing, or management practices that emerge from these studies are referred to by many names such as: total quality design/development, concurrent engineering, etc.. A collection of structured methods developed and used in these practices is called Quality Function Deployment (QFD), among which there is one popularized method called the House of Quality (HoQ). The popularity and success of QFD and of the HoQ is detailed in many recent publications (Akao, 1990; Claussing, 1993; Gevirtz, 1994; King, 1989).

This paper focuses on the role of computational support for QFD techniques. Such support received significant attention in the last few years as displayed in a recent survey of software tools for quality assurance or control, documenting 526 software packages (Brecka, 1995). By and large, these software tools provide means for data input, storage, manipulation, and presentation which are of practical importance. Furthermore, the development of these tools is not trivial, facing with complex issues of usability, interface design, and functionality coverage that must be solved to compete in this software market. Nevertheless, this paper emphasizes opportunities for computational services beyond computerizing the manual work on QFD; services whose potential contribution to QFD are recognized but presently unavailable (Oakland, 1989). This emphasis is analogous to 3D CAD systems with visualization, interference checking, volume calculation, and other facilities that went beyond the mechanization of 2D drafting. We intend to explore techniques, "intelligent" and "dumb," that can extend the scope of current QFD including the future reuse of designs developed with it. These additional techniques will be borrowed from the high-end KIC technologies as well as from other sources. Independent of which technique is explored, a commitment is made to maintain the practical relevance of the evolving QFD tools.

Since the techniques themselves they do not have knowledge and do not design on their own, we have to answer why a computational support for QFD constitutes KIC. The answer is simple: once the techniques are used they start storing significant amount of knowledge about design alternatives, design decisions, priorities, competitors data, etc. Given the right organization, management, and retrieval facilities, these techniques could provide practitioners with abundance amount of information for making more informed use of manual QFD tools.

The remainder of the paper is organized as follows. Section 2 introduces the QFD techniques we would like to support. Section 3 discusses the relation between computations and QFD. Section 4 discusses how an implementation of a computational QFD tool is approached and Section 5 concludes.

2 TOTAL QUALITY DESIGN

Quality means meeting the requirements whether stated or implied. Thus, quality design involves collecting and understanding of design requirements and producing products that meet them. However, what, for example, are the requirements of a design that is completely new? or, who are the customers of new designs from whom requirements are to be solicited. These questions and others critical issues pertaining to design processes (Reich et al, 1992) are outside the scope of this paper. Since we focus on providing support for *existing effective manual techniques in whatever ways they are used today* we can defer the treatment of these issues.

2.1 Seven (new) management tools

There are many views about what total quality design is; yet, there is a consensus about the central role of one item: *information quality* (Claussing, 1993; Gevirtz, 1994; Ohno and Mito, 1988). QFD tools are aimed at eliciting information from various sources including customers, engineers, and past product performance and organizing them in various ways so that they can be used for planning and design. QFD tools are built to represent information in simple *graphical models* that are very easy to comprehend.

An early set of tools, referred to as the seven (old) quality control tools, assisted in statistical quality control (SQC). It includes (Ishikawa, 1982): check sheets, Pareto charts, histograms, cause-and-effect diagrams, scatter charts, control charts and bar graphs. They all have proved useful for their intended tasks. As it became apparent over the years, SQC could control the production process but *not improve design quality.* Consequently, new requirements for QFD tools were formulated to include three categories of provisions (Mizuno, 1988).

1. *Expressiveness.*
   a) Supporting the processing, representation, and communication of verbal information.
   b) Providing a structured way to organize information.
2. *Adaptability.*
   a) Being flexible to accommodate the diversity of ways people may wish to use the tools.
3. *Communication.*
   a) Being simple enough to use and understand so that they can serve as communication tools between everyone in the company.
   b) Eliminating failures or slippage by identifying all factors contributing to an issue.
   c) Easing the information exchange between process participants.
   d) Assisting in communicating and disseminating information in the organization.
   e) Encouraging the use and storage of raw, unfiltered, expressed information for future reuse.

f) Fostering ideas generation and their clear presentation.

These categories clearly show the critical role of information communication on quality design. We return to these three categories later when classifying the limitations of QFD tools and when identifying the benefits from computational support tools.

We will refer to this set of requirements as *QFD requirements.* A collection of QFD techniques, called the seven (new) management tools, evolved based on these requirements (Oakland, 1989; Mizuno, 1988).

1. *KJ method (affinity diagram)*, which is a brainstorming tool, clarifies important concepts by collecting and organizing diverse verbal data (e.g., ideas, concepts, issues, opinions, etc.) into groupings. The results can be fed into the interrelation, tree, or matrix diagrams. A by-product of using this method is the establishment of a common terminology for a particular product. Simple problems or those requiring quick solutions do not lend themselves to the use of this technique.
2. *(Inter)Relation diagram* clarifies the logical relationships between concepts related to the product.
3. *Tree (systematic) diagram* is used to decompose a problem into its contributing factors. A problem can be a goal to attain or a product and the factors can be actions or components, respectively.
4. *Matrix diagram* is the central tool of the seven. It clarifies the relationships between different facets of a design such as functions, tasks, or requirements, and identifies their relative importance. There are several types of matrices depending on the number of facets that are used. A L-shaped matrix involves two, and a X-shaped matrix four facets. A C-shaped matrix involves three facets in a 3D space and is therefore hard to conceptualize or construct on a piece of paper.
5. *Matrix data analysis* calculates and displays tabular data in a graphical form. Its major application is in the analysis of matrix facets correlations by principal components analysis.
6. *Process decision program chart (PDPC)* is used to uncover and display all the events (expected or undesired) that can happen when implementing a plan. Events are listed with their possible outcomes and possible countermeasures if required. This tool can borrow input from a tree diagram that details plans and proceed with the use of failure mode and effect analysis (FMEA).
7. *Arrow diagram* is used to schedule or select the most appropriate plan. It is similar to Gantt charts and can be used for critical path analysis and for other project planning purposes.

The relationships between the new tools, the old tools, and their role in various product development stages is shown in Figure 1. The first two tools are used in the planning stage where information is elicited from various sources, the next three are used for searching for solutions, and the last two for controlling the implementation of the plans. Clearly, other tools, such as Pugh's concept selection method (Pugh, 1981), can join this group and the existing tools may evolve to support new practical demands. The tools are very simple and not necessarily new. Their strength lies in their usability and understandability. As seen from Figure 1, the tools are best used as a group.

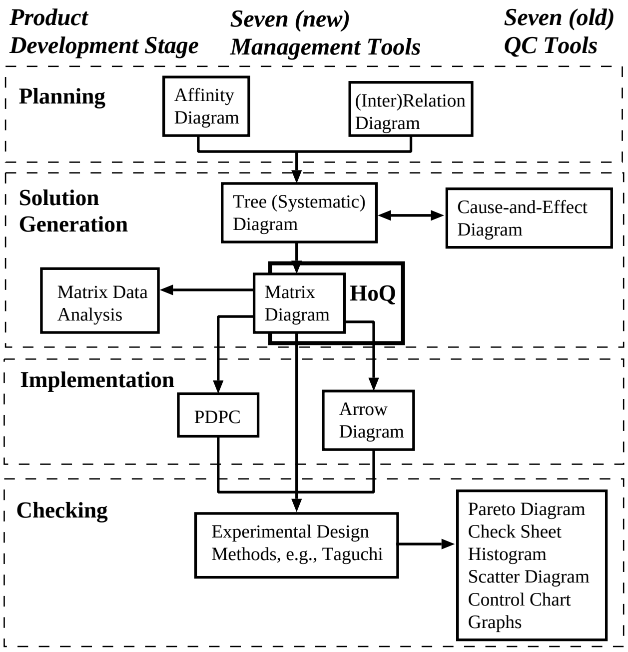

Figure 1 Relationships between the 7 new and old quality management tools.

2.2 The House of Quality (HoQ)

The matrix technique has received much attention and some developed it as a central tool in their approach to quality design, referred to as the House of Quality (HoQ) due to its layout (see Figure 2) (Akao, 1990; Claussing, 1993; Gevirtz, 1994; King, 1989). The HoQ has enjoyed significant success in industry as a design tool especially for relating customer requirements to product measurable attributes. Its use involves several steps (whose numbers refer to activities associated with the room numbers in Figure 2):
1) *Extracting customers' voice.* This step extracts users' requirements of the design. This is done through surveys, interviews, customer complaints data, repairs data, etc.. The affinity diagram and the interrelation diagram can are used as tools in this step.
2) *Performing competitive analysis of products with respect to customers' voice.* This step rates and compares different available products with respect to the requirements extracted in the first step. It sets up the product goals with respect to customers voice and calculates the relative weight of customers requirements.
3) *Expressing the engineers' voice.* This step involves defining product attributes that translate the abstract customers' requirements to measurable values.
4) *Uncovering "how" relations.* This step identifies how product measurable attributes determine customers' requirements.
5) *Performing technical comparison.* This step evaluates the same designs that were used in step 2 according to the measurable product attributes. Part of this step also involves checking consistency between the competitive analysis, correlations, and technical comparison values.
6) *Uncovering trade-offs.* This step identifies the trade-offs between the product attributes.
7) *Defining product goals.* The competitive evaluation, technical comparison and the correlations between the the customers' requirements and product attributes determine the

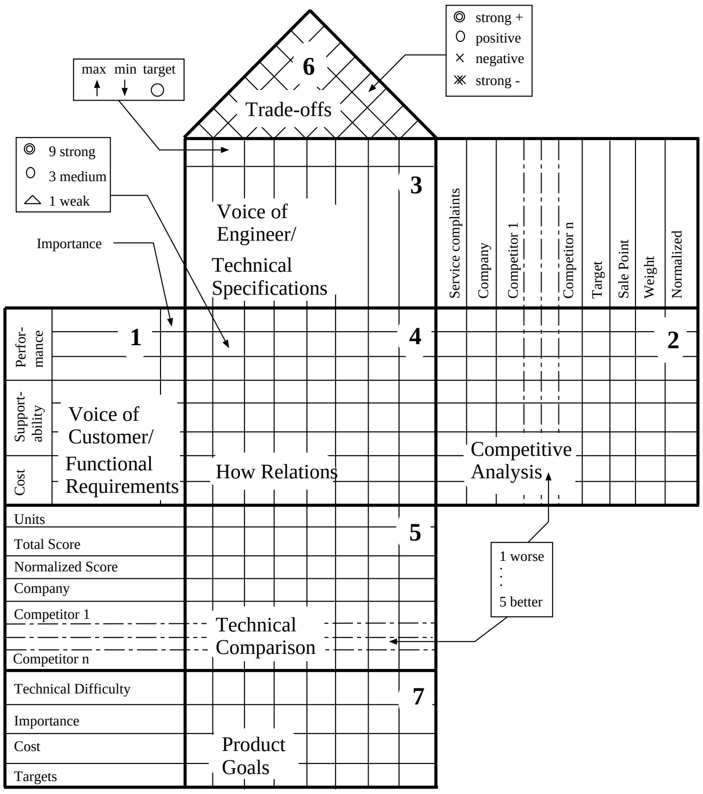

Figure 2 The House of Quality.

target goals of the design in terms of values of the product attributes. These targets become the input to subsequent QFD stages.

The creation of the HoQ involves significant domain expertise. Both empirical knowledge originating from previous projects is required as well as technical knowledge; they come into play mainly in creating rooms 3, 4, and 6. When improving existing products, empirical knowledge is critical and when designing new products, the role of technical knowledge increases.

One of the changes to Japanese practices that was introduced by several QFD consultants (King, 1989; Claussing, 1993) is Pugh's concept selection method (Pugh, 1981) for creating preliminary designs. While the regular QFD methods are geared towards improving design reliability and reducing cost, Pugh's method is geared towards encouraging alternative perspectives.

2.3 Limitations of QFD tools

By and large, most QFD implementations use paper and pencil as tools. Manual paper techniques have several general limitations that can be classified into four categories:
1) The *effort* category deals with the ease of handling a tool including issues such as: size, editing, maintaining consistency, and performing various computations.
Size. It is difficult to handle large amounts of data on a paper. Tree diagrams and matrix diagrams can become too large to cope with when the number of items grows and matrix diagrams for large projects can easily have hundreds or thousands of items (Claussing,

1993). Users of existing tools attempt to cope with this limitation by aggregating issues to limit their number.
Editing. Once the results of using the tools are displayed, it will take significant effort to make any revision such as inserting another level in a tree diagram or adding additional customer requirements in room 1 of a matrix diagram since this may involve a complete rewrite of the paper layout.
Consistency. A paper "suffers" any information written on it. It is the role of its users to make sure that the information is correct. As an example of suspected inconsistency consider a product that is rated 5 (in room 2) on the users' requirement `compact size` and rated 1 (in room 5) on the product attribute `volume`, even though they are highly correlated (in room 4). In addition, when using QFD tools for product design, various concepts are reused by different QFD tools. Users can easily fail to relate similar concepts in different graphical models thus lose potentially useful information or introduce inconsistencies into the graphical models.
Computations. Similar to the consistency item, the users are in charge of all the calculations involved in using paper methods. For example, calculations can become a burden especially if one wishes to investigate the sensitivity of the results to various ratings by the tool users.
2) The *expressiveness* category deals with the nature of information that a QFD tool accepts including issues such as the type of information it can encode and its ability to capture rich design rational.
Type of information. The existing structure of the HoQ and the related QFD techniques restrict the nature of information that can be coded. For example, the house permits only several quantitative values to be used as evaluations or correlations. However, in some situations, richer and more precise correlations can be established based on parametric analysis or theoretical relations. It may be desirable to adapt a tool to make use of such information when it becomes available.
Rational. QFD tools force their users to structure the decision process and assist in keeping records of major decisions and parts of the processes that generated them. Nevertheless, if we look at QFD graphical models, there is significant amount of information that is not recoverable from them. For example, since the data displayed by many of the QFD tools is a result of group processes, the individual positions are missing and the reasons that led to existing choices are unavailable.

Paper documents are linear. Thus, any information that might have been useful to associate with information described in QFD graphical models cannot be kept where it best belong. Thus, we may have difficulties in understanding parts of these models.
3) The *flexibility/adaptability* category contains issues pertaining to the ease of modifying a technique to suite changing needs.
Adaptability. There are two major ways in which a tool can be adapted. In one way an existing tool or graphical model can be modified by modifying or removing some entries or even by modifying the interface. In this way the model serves as a *prototype* while the tool itself remain intact. QFD tools have been adapted significantly in this way considering, for example, the diversity of ways in which Japanese have used them (e.g., more that 80 types of matrices (King, 1989)).

Observing this diversity and considering that some tools are "too" flexible for novices (Mizuno, 1988), it becomes useful to include guidelines or templates that users can retrieve and adapt based on past uses of a tool that are similar to the design case at hand. This may be difficult with paper methods, but is doable with computational tools. In essence, previous cases of using QFD tools serve as *prototypes* for new usage and the new guidelines or templates provide a *language* for writing new graphical models specific to a particular kind of problems. These two roles, a prototype and a language, are borrowed from n-dim (Levy et al, 1993) and will be elaborated later.

4) *Communication.* The use of QFD techniques forces having good face-to-face (synchronous) communication practices because most of the tools are used in group settings. Sometimes, in the dynamics of group discussions, it may not be easy to distinguish between different positions or to observe the similarities between others thus some information may be lost. Also, in the rational item we observed that significant information is lost when QFD models are used in asynchronous manner, whether for communication among members of the current project or for disseminating information into the organization.

3 QFD AND COMPUTING

3.1 Benefits of computational support for QFD

It is clear that computer tools can alleviate some of the aforementioned difficulties or limitations of QFD tools. The key issue is which tools to use and how to build them such that they maintain the usefulness of the original QFD tools and do not induce restrictiveness over users (Chu and Elam, 1990). As a guideline, any computational QFD tool must as least satisfy QFD requirements and, moreover, demonstrate an advantage over manual techniques with respect to satisfying them. Let us address the provision of computational support for each of the difficulties in turn.

Size. Computer tools can be used to construct large models as desired subject only to computer hardware limitations. They can provide facilities to focus on parts of the data while ignoring the rest, or for aggregating data. Note that it is not clear whether large models are practically useful or how to manage them effectively. However, computer tools by virtue of their flexibility allow users to experiment and discover themselves what is the modeling style that suites them best. In parallel, empirical work is needed for establishing recommendations about reasonable sizes of the various models.

Editing. Computer tools can easily support modifications including significant changes to layout and content of graphical models. With the right information management, some consistency issues can be dealt with during such revisions. This issue also relates to reusability; computer tools allow to easily copy portions of graphical models to be used for new product developments.

Consistency. Computerized techniques can provide support for detecting conflicts and checking consistency in a graphical model, especially in the HoQ. Computer tools allow to better maintain consistency across different QFD tools that are used for the same product development. This may involve checking the existing terminology and using its concepts or relating new concepts to existing ones. Such consistency also improves the understandability and communication of information represented with graphical models.

Computations. Computational tools can easily support the computations required in executing QFD tools. For example, automating the computations in the HoQ can support quick sensitivity analysis of the target goals in room 7 with respect to the ratings in room

3. In addition, computational support for QFD introduces opportunities not presently possible for invoking other tools during modeling. Later we detail some of the tools that may be used.

Expressiveness. Computer tools allow to associate various kinds of information or models with data displayed by the tools. Thus, for example, correlations in room 4 or 6 of the HoQ can be derived explicitly from equations or previous information stored in a database or can be expressed as formulae.

Rational. Computational tools based on hypertext models can be introduced to attach information to parts of graphical models. This information can be simple or arbitrary complex such as complete rational models developed in IBIS-like languages (e.g., GIBIS (Conklin and Begeman, 1988)).

Adaptability. In discussing the limitations of adapting paper methods and in the revisions item above we have already mentioned that computational tools significantly ease the adaptation of QFD tools. For example, since previous uses of the tools can be stored and categorized, different templates can be created that will be used for future relevant cases. Also, computer tools present opportunities for designing interfaces and functionalities that can be adapted to suite personal preferences or practical needs of different users.

Communication. Current computing technology can be used to remove the same-place restriction on communication. Conferencing tools allow participants from different locations to communicate. The rational item above deals with methods for improving the communication of information asynchronously. Richer models with significant context must be used to support such different-place different-time communication (Subrahmanian et al, 1993a).

3.2 The role of general graph representations

From the requirements of QFD tools and the potential benefits from computational support, it follows that a computational QFD (CQFD) tool needs to: (1) provide facilities for storing, organizing, and retrieving graph-based information models; (2) allow attaching computational services to the graphical models; (3) provide support for adding expressive power to the available techniques; (4) include graphical user interfaces that support easy entry and manipulation of data and that can be adapted easily to support different users' preferences or different QFD tools; and (5) have facilities for fostering idea generation.

Among the critical choices when designing a CQFD is the selection of the information representation scheme. If we look at the output of QFD tools, they all have graphical representations that we call *graphical models.* Moreover, all but the matrix techniques employ some *graph* representation and matrix techniques can easily be represented as graphs with different *presentation* schemes such as the one in Figure 2.

The building blocks of these graphical models are objects that designate domain concepts. The objects have labels whose meaning is most often agreed upon in group discussions. The links (or relations) are also domain concepts with labels whose meaning was built into the particular QFD tool that is being used. The type (or role) of concepts, whether objects or links, is also built into the QFD tool. For example, a particular product assembly can appear in a graph depicting part decomposition developed using the tree diagram tool; in this case, the role of the object representing the assembly would be the product structure. Subsequently, the same assembly can appear in one PDPC diagram detailing its functioning and in another detailing its manufacturing process. In each of these diagrams the assembly would have a different role.

From the example, it is evident that the same concept can appear in several graphical models. Thus, a concept will have many links connected that originated from the use of different QFD tools. If we lay down all the concepts with their relationships as recorded in the various models we will get a big general graph with many node and link types. This is similar to the notion of "flat space" borrowed from n-dim that will be discussed later. Thus, any individual graphical model contains a subset of the objects with links that are interpreted to have a particular role or type depending on the model they are in. The model provides the context for the interpretation.

A graph is selected for information representation due to three reasons. First, it naturally emerges as a way of conceptualizing the graphical models of manual QFD tools. This supports an evolutionary adjustment of users from the manual to the computational tools.

Second, there are various kinds of graph representations associated with computer tools that may become readily available for users of graph-based computational techniques. Two particular types of graph-based models we refer to are: conceptual structures (Sowa, 1984) and influence diagrams (Howard and Matheson, 1983).

Conceptual structures are a graphic system for representing concepts and relations that is general as predicate calculus. There have been studies on the use of conceptual graphs for various CAD-related topics such as: mapping the enterprise information language EXPRESS (Wermelinger and Bejan, 1993), modeling object features and constraints for CAD (Salomons et al, 1994), and modeling the knowledge interchange format KIF (Sowa, 1993). In contrast to these and other capabilities, conceptual graphs do not represent uncertainties well. On the other hand, *influence diagrams* are graph-based models for representing probabilistic knowledge for probabilistic inference. They are extensions of Bayesian networks and have been used in artificial intelligence, decision analysis and statistics. The use of a graph representation can ease the use of conceptual graphs and influence diagrams as specialized models, as well as other graph-based modeling tools.

The third reason for selecting graph representation is its correspondence with two theories of design: the mathematical theory GDT (Yoshikawa, 1981; Tomiyama, 1994) and the empirical theory of design as emerging from the n-dim project (Konda et al, 1992; Levy et al, 1993; Subrahmanian et al, 1993b).

According to GDT, knowledge is represented by a topological structure consisting of objects and their relationships. Using this and few other assumptions, GDT proves interesting properties about design. Over the years, the theory evolved into an integrated modeling environment based on the concept of metamodel (Tomiyama, 1994; Yoshikawa et al, 1994) A metamodel is an ontology of concepts and relationships about physical phenomena. The metamodel is represented as a graph and is used to generate models of particular phenomena for simulations.

n-dim is based on empirical studies of designers contending that design is a continual negotiation of constraints, terminology and trade-offs for the creation of a shared understanding of the design process and product (Subrahmanian et al, 1993a). Designers exchange information expressed in different representational forms (e.g., pictures, text), in different media (e.g., paper, electronic), and in different modes (e.g., formal, informal). In manipulating design information designers use a variety of models (Subrahmanian et al, 1993b). n-dim is implemented under the assumption that a general graph-based modeling environment can capture significant portion of the models used by designers, where models are collections of objects and relations. Thus, the set of models is a set of multiple potentially overlapping classifications over the universe of objects. As this set grows, it better approximates the topological structure of knowledge hypothesized by GDT (Reich, 1995).

4 IMPLEMENTATION

In the previous section we mentioned *n*-dim (Levy et al, 1993) as a project whose underlying ideas have influenced the present work. This section briefly reviews some principal ideas of *n*-dim, argues that *n*-dim is an ideal system within which to implement CQFD tools, and describes the architecture of an *n*-dim-based CQFD tool.

4.1 A short *n*-dim overview

n-dim is a system developed to support collaborative design. It has been developed following, and in parallel to, empirical data from collaborative engineering projects (Subrahmanian, 1992). It is designed to have the functional facilities for supporting such work and its implementation is designed to really scale up by paying careful attention to the architecture design (e.g., a distributed layered architecture) and the software development languages (e.g., in-house development of a prototype-based toolkit and language). In reviewing *n*-dim we will list in parentheses the QFD requirement that is supported by the particular *n*-dim feature.

Flat Space of objects and models. An *n*-dim object can be anything that can be stored electronically. The world according to *n*-dim is conceptually flat: objects do not contain other objects. There are two categories of objects in *n*-dim: atomic and structured where the latter are built out of other atomic or structured objects. A model is the primary type of structured object: a collection of objects and their *links* or relationships, which are themselves objects (satisfies **(1b,3a)**).

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 and evolve over time. *n*-dim provides a set of synchronous and asynchronous communication mechanisms for doing so.

Once constructed, models in *n*-dim become regular objects; thus, the same object or model may participate in many other models. The simplicity and generality of model structures enable the capture of rich context of a given object.

Roles of a model. Models play two primary roles in *n*-dim: instance/prototype and language. In its role as a prototype, a model may be copied to serve as initial conceptualization 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. A model viewed as a modeling language can be thought of as a grammar.

Clearly modeling languages can be created for generating any syntactically correct graph-based model including the models of the 7 management tools and the HoQ. The conceptual separation between models' structure and presentation allow viewing the models in their natural way (e.g., viewing the HoQ as in Figure 2 and not as its underlying graph). Similarly, modeling languages can be constructed for generating conceptual graphs, influence diagrams, or the metamodel that were discussed before.

Communication. n-dim provides a variety of communication mechanisms: synchronous and asynchronous. The synchronous is not conceptually difficult compared to the asynchronous. Asynchronous communication is supported by maintaining the context of discussions in addition to supporting history or rational maintenance modeling languages like IBIS **(3c, 3d)**.

For example, when a group negotiates over an issue, a member of the group can alter a model and make it persistent and exchangeable by *publishing* it. Another member can then follow by copying the model, making revisions and publishing it again. n-dim maintains pedigree of information in revision models through which the evolution of the model can be easily traced.

External tool encapsulation. n-dim allows to encapsulate external tools written in diverse programming languages and use them in models. One such tool is a natural language processing (NLP) tool that can build terminological structures from a corpora of text **(1a, 3c)** (Reich et al, 1993). This tool can be of significant help for operating the affinity diagram tool. Other candidate tools for encapsulation include inference engines for conceptual structures or influence diagrams, or simulations of qualitative physics graphs derived from the metamodel.

n-dim has other facilities such as searching for information in models **(3b,3d)**, ability to incorporate any information that can be transferred to electronic media in its models **(3e)**, and synchronous communication means **(3c)**. n-dim is made to sustain evolutionary development and adaptation by different users **(2a)**, although operationalizing it is a challenge for n-dim as well as for any CQFD tool. All the above leaves **(3f)** as the only requirement that n-dim does not currently address. This can be improved by incorporating groupware (Nunamaker et al, 1988) or other creativity-fostering tools developed specifically for these purposes.

4.2 An architecture of a CQFD tool

Figure 3 shows the architecture layers of an n-dim-based CQFD tool. All but the top layer are n-dim layers (Levy et al, 1993) demonstrating the great advantage of using n-dim as the implementation vehicle. With little attention to lower layers and only concentrating on the first and maybe second layers, we can obtain a flexible tool that runs on a heterogeneous environment and provides additional functionalities such as communication or distribution for free.

This should not be construed as if n-dim is the key to building CQFD. The key issue is how to make a computational tool usable, practically effective and appealing so it is adopted for use by present users of QFD techniques, or that it is adopted as a QFD tool

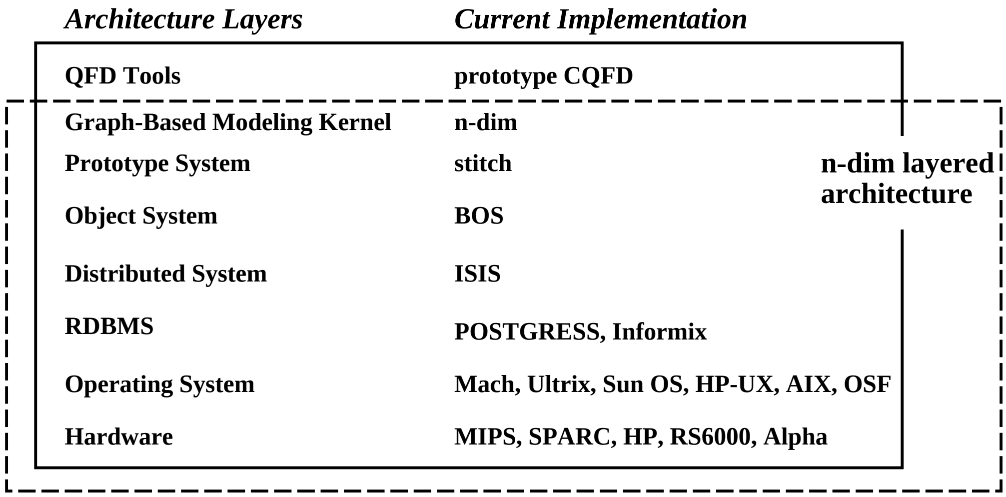

Figure 3 An architecture of CQFD.

by newcomers to QFD and is used to their advantage. *n*-dim seems to be a perfect match for the needs of CQFD, however, other implementations may be possible as well.

4.3 Using a CQFD tool

A CQFD tool provides a way to integrate all QFD tools with other computer services together. In the planning stage (see Figure 1), various textual data can be analyzed by an NLP tool to assist in the creation of terminology by the affinity diagram. The clusters of the terms in the affinity diagram can be presented as in the manual QFD technique with an underlying representation of a tree. Interrelation diagrams can be constructed by incorporating the objects from the affinity diagram, by searching for similar objects, products, or physical phenomena in the system's database and incorporating them, or by generating new objects.

In the solution generation stage, products that are connected to similar terminology or interrelation diagrams can be retrieved, or other models that may be relevant. If such objects do not exist, new tree diagrams can be created. The collection of terms and other objects created, as well as others that may be retrieved from the database can be used for constructing the HoQ matrix. Here, similar to the use of the affinity diagram, the user interacts with a presentation method different than the underlying representation. An existing template is retrieved that already includes the various calculations associated with the HoQ. These can calculate output values whenever their input is complete or modified.

In the implementation stage, the arrow diagrams and the PDPC diagrams are created using their modeling languages. During this whole process, the user can leave a particular model and inspect or generate other models. The system meanwhile updates the underlying flat graph representation with new objects and links. We already mentioned that the database (or the graph) can be searched for objects, but moreover, if an object already exist in the graph, the user can inspect the models in which the object participates to learn more about it and not merely use it in subsequent models. This provides rich context for understanding previous products.

The underlying graph can be used also for mapping between existing data to new models for performing computational services. This mapping may be required because n-dim or CQFD tool models can be informal with open ended meaning, whereas, models for computational services must have concise interpretation.

Finally, the user can observe during inspection of several products that their designs used particular types or alterations of QFD tools. Thus, the user can evolve new modeling languages for this specific type of products. In the same way, new QFD tools may be created from informal models that previous users created and that were found practically useful.

4.4 Related work

There has been little work done on supporting QFD with techniques beyond the spreadsheet level. Several studies have dealt with this issue to a limited extent. One system, called Design Scribe (Bradley and Agogino, 1991), is a connected collection of several graphical models including: a simple form of QFD, functional diagrams, form-function decomposition diagrams, morphological matrices, concept evolution diagrams, and parts of methodical design diagrams. The system is implemented as a hypercard and is directed towards recording these models as a form of design history. Another hypertext implementation merely provides a usable interface to the technique (Wolfe, 1994), although its developer discussed attaching several other computational services.

A third study deals with incorporating mathematical models in the roof of the HoQ (Ramaswamy and Ulrich, 1993). Much more complex models can be incorporated such as qualitative physics knowledge and simulations. The critical issue is whether such techniques restrict the way users currently use the HoQ; thus, the utility of these computational techniques is an empirical question. That study also advocates for using generalized graphs for representing both the HoQ and the mathematical model information.

5 SUMMARY

This paper establishes that the development of computational support tools directed primarily towards supporting manual information intensive tasks is an important undertaking for KIC research. Specifically, QFD as a collection of design support tools can enjoy from computational support beyond the spreadsheet-like level. Computational tools can ease the effort of using QFD tools, improve the expressiveness of information they record, improve their adaptability to new needs, and better support the communication of QFD graphical models.

Through the use of an underlying graph representation, QFD graphical models can better share information between themselves and with other computational services developed on graph structures. In this way, CQFD can easily incorporate IBIS-like design rationale models and interface with computational services developed for conceptual graphs, influence diagrams, or qualitative physics models.

The paper describes an architecture based on n-dim that provides the infrastructure for embedding various graphically-oriented QFD tools. The use of n-dim adds many additional attractive properties such as a distributed environment, information management in a database, and communication facilities.

We emphasized that many individuals and organizations adapted QFD methods to suite their own needs and preferences and that the usability and practical utility of CQFD tools are the critical aspects in their development. Presently, we are in the process of implementing a simple CQFD tool with an IBIS-like rational capture method. This tool will be used in a design project course and will provide insight about the usability of CQFD tools before embarking on a full integration with *n*-dim. From experience with other QFD tools, we already know that even such tool leaves much information unrecorded. The use of *n*-dim will allow us to incrementally improve our CQFD design.

If we view this document as a conceptual design of a CQFD tool, then we should attempt to cast this design *within* a QFD framework. This involves using QFD tools to organize "customers requirements" from the literature and potential users and using the 7 tools, the HoQ, and other tools such as IBIS-based design rational capture, to translate these requirements into a design. This process will check the validity of the present conceptual design and will determine the priorities for satisfying the goals or requirements in developing a CQFD tool as will be set in room 7 of the HoQ.

6 ACKNOWLEDGMENTS

The *n*-dim project and its other members, Robert Coyne, Douglas Cunningham, Allen Dutoit, Eric Gardner, Suresh Konda, Sean Levy, Ira Monarch, Robert Patrick, Mike Terk, Eswaran Subrahmanian, Anish Srivastava, Mark Thomas, and Art Westerberg, have been a source of insight for the ideas presented in this paper and will continue to be such source whenever truly practical applications are sought.

7 REFERENCES

Akao, Y., editor (1990). *Quality Function Deployment*, Productivity Press, Cambridge, MA. Original Japanese version from 1988.

Bradley, S. R. and Agogino, A. M. (1991). "Design capture and information management for concurrent design." *International Journal of Systems Automation: Research & Applications*, 1(2):117–141.

Brecka, J. (1995). "Software packages: Quality Progress' 12th annual QA/QC software directory." *Quality Progress*, 28(3):25–119.

Chu, P. C. and Elam, J. J. (1990). "Induced system restrictiveness: An experimental demonstration." *IEEE Transaction on Systems, Man, and Cybernetics*, 20(1):195–201.

Claussing, D. (1993). *Total Quality Development*, ASME Press, New York, NY.

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.

Deming, W. E. (1993). *The New Economics*, MIT Center for Advanced Engineering Study, Cambridge, MA.

Ealey, L. A. (1994). *Quality by Design: Taguchi Methods and US Industry*, The American Supplier Institute Press, Burr Ridge, IL, 2nd edition.

Fenves, S. J., Garrett, J. H. J., and Hakim, M. M. (1994). "Representation and processing of design standards: A bifurcation between research and practice." In *Proceedings* *of the 1994 Structures Congress (Atlanta, GA)*, New York, NY, American Society of Civil Engineers.

Gevirtz, C. D. (1994). *Developing New Products with TQM*, McGraw-Hill, New York, NY.

Howard, R. A. and Matheson, J. E., editors (1983). *Readings on The Principles and Applications of Decision Analysis*, Strategic Decisions Group, Menlo Park, CA.

Ishikawa, K. (1982). *Guide to Quality Control*, Asian Productivity Organization, White Plains, NY, 2nd rev. edition. Original Japanese edition by JUSE from 1968.

King, B. (1989). *Better Designs in Half the Time: Implementing QFD Quality Function Deployment in America*, GOAL/QPC, Methuen, MA, 3rd edition.

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.

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.

Liker, J. K., Fleischer, M., Nagamachi, M., and Zonnevylle, M. S. (1992). "Designers and their machines: CAD use and support in the US and Japan." *Communications of The ACM*, 35(2):77–95.

Mizuno, S., editor (1988). *Management for Quality Improvement: The Seven New QC Tools*, Produtivity Press, Cambridge, MA. Original Japanese edition by JUSE from 1979.

Nunamaker, J. F., Applegate, L. M., and Konsynski, B. R. (1988). "Computer-aided deliberation: Model management and group decision support." *Operations Research*, 36(6):826–848.

Oakland, J. S. (1989). *Total Quality Management*, Butterworth, Oxford, England.

Ohno, T. and Mito, S. (1988). *Just-in-Time for Today and Tomorrow*, Productivity Press, Cambridge, MA. Original Japanese edition by Diamond Inc. from 1986.

Pugh, S. (1981). "Concept selection- a method that works." In *Proceedings of the International Conference on Engineering Design (ICED-81)*, pages 497–506, Heurista, Zurich.

Ramaswamy, R. and Ulrich, K. (1993). "Augmenting the house of quality with engineering models." *Research in Engineering Design*, 5:70–79.

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.

Reich, Y., Konda, S., Levy, S. N., Monarch, I., and Subrahmanian, E. (1993). "New roles for machine learning in design." *Artificial Intelligence in Engineering*, 8(3):165–181.

Reich, Y. (1994). "What is wrong with CAE and can it be fixed." In *Preprints of Bridging the Generations: An International Workshop on the Future Directions of Computer-Aided Engineering*, Pittsburgh, PA, Department of Civil Engineering, Carnegie Mellon University.

Reich, Y. (1995). "A critical review of General Design Theory." *Research in Engineering Design*, 7(1):1–18.

Salomons, O. W., van Slooten, F., de Koning, G. W. F., van Houten, F. J. A. M., and Kals, H. J. J. (1994). “Conceptual graphs in CAD.” *Annals of the CIRP*, 43(1):125–128.

Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA.

Sowa, J. F. (1993). “Relating diagrams to logic.” In Mineau, G. W., Moulin, B., and Sowa, J. F., editors, *Conceptual Graphs for Knowledge Representation, Proceedings of the First International Conference on Conceptual Structures (ICCS'93), (Quebec City, Canada)*, pages 1–35, Berlin, Springer-Verlag.

Subrahmanian, E., Coyne, R., Konda, S. L., Levy, S. N., Martin, R., Monarch, I. A., Reich, Y., and Westerberg, A. W. (1993). “Support system for different-time different-place collaboration for concurrent engineering.” In *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises (WET ICE)*, pages 187–191, Los Alamitos, CA, IEEE Computer Society Press.

Subrahmanian, E., Konda, S. L., Levy, S. N., Reich, Y., Westerberg, A. W., and Monarch, I. A. (1993). “Equations aren't enough: Informal modeling in design.” *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, 7(4):257–274.

Subrahmanian, E. (1992). “Notes on empirical studies of engineering tasks and environments, invited position paper.” In *NSF Workshop on Information Capture and Access in Engineering Design Environments (Ithaca, NY)*, pages 567–578.

Tomiyama, T. (1994). “From General Design Theory to knowledge intensive engineering.” *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 8(4):319–333.

Umeda, Y., Tomiyama, T., and Yoshikawa, H. (1991). “A design methodology for a self-maintained machine.” In *Design Theory and Methodology–DTM'91 (Miami, FL)*, pages 143–150, New York, NY, The American Society of Mechanical Engineers.

Wermelinger, M. and Bejan, A. (1993). “Conceptual structures for modeling in CIM.” In Mineau, G. W., Moulin, B., and Sowa, J. F., editors, *Conceptual Graphs for Knowledge Representation, Proceedings of the First International Conference on Conceptual Structures (ICCS'93), (Quebec City, Canada)*, pages 345–360, Berlin, Springer-Verlag.

Wolfe, M. (1994). “Development of the city of quality: a hypertext-based group decision support system for quality function deployment.” *Decision Support Systems*, 11(3):299–318.

Yoshikawa, H., Tomiyama, T., Kiriyama, T., and Umeda, Y. (1994). “An integrated modelling environment using the metamodel.” *Annals of the CIRP*, 43(1):121–124.

Yoshikawa, H. (1981). “General Design Theory and a CAD system.” In Sata, T. and Warman, E., editors, *Man-Machine Communication in CAD/CAM, Proceedings of The IFIP WG5.2-5.3 Working Conference 1980 (Tokyo)*, pages 35–57, North-Holland, Amsterdam.

8 BIOGRAPHY

Dr. Yoram Reich is a Senior Lecturer in the Department of Solid Mechanics, Materials and Structures, Faculty of Engineering, Tel Aviv University, Israel. He received his BSc (Summa Cum Laude) and MSc (Magna Cum Laude) in Mechanical Engineering from Tel

Aviv University in 1980 and 1984, respectively. Before obtaining the PhD degree in Civil Engineering from Carnegie Mellon University in 1991, he practiced engineering design for over 7 years in the audio, structures and marine industries.

Yoram Reich has authored or co-authored of over 50 papers and is a member of the editorial board of the Journal of Artificial Intelligence in Engineering. His research focuses on several interrelated topics: computational support for collaboration and participation in design, machine learning and knowledge acquisition techniques for engineering applications, quality design, design theories, and research methodology. He recently edited a special issue of the journal AI EDAM on the last subject.