Building Agility for Developing Agile Design Information Systems

Yoram Reich† Suresh Konda§, Eswaran Subrahmanian‡, Douglas Cunningham‡, Allen Dutoit§, Robert Patrick‡, Mark Thomas‡, Arthur W. Westerberg‡and the *n*-dim group‡

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

§Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, email: slk@sei.cmu.edu, Tel: 1-412-268-5221, Fax: 1-412-268-5229

‡Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA 15213, USA, email: sub@globe.edrc.cmu.edu, Tel: 1-412-268-5221, Fax: 1-412-268-5229

To Appear in *Research in Engineering Design*

Abstract: Agile manufacturing relies heavily on the quality of information that organizations have and on their ability to organize and reuse it. Constant inflow of information and knowledge is the fuel of agile manufacturing. In the process of forming virtual enterprises, these new organizations have to be equipped with information systems that integrate their present legacy technology and improve upon it. In order to support the quick formation of virtual organizations, one must have the ability to develop such systems quickly. Over the past years we have evolved, through collaborative projects with industry an approach composed of methods and an information infrastructure called *n*-dim that improves the ability of becoming agile manufacturers of information systems, by responding quickly to information needs of new and evolving organizations. Following an analysis of the requirements of information systems for agile design, we discuss this approach; describe some of the infrastructure features and present several examples of simple applications that illustrate them. We summarize by discussing the advantages and limitations of our approach.

Index Terms: Information Infrastructure, Cooperative Design, Information Modeling, Information System Design, Agile Design Information System, Computer-Supported Cooperative Work

1 Introduction

Agility is a new perspective about the relationships between manufacturers and customers: a relationship of enrichment. This relationships shifts the focus of business from selling products to providing services and complete solutions (Goldman et al, 1995) in a dynamic environment. This perspective has significant ramifications for the way manufacturing organizations need to operate. They must be capable of (1) quickly detecting changing markets; (2) rapidly learning to take advantage of these market changes; (3) detecting new techniques, adapting them to the enterprise culture, assimilating them into the enterprise while maintaining their spirit, and using them effectively; (4) meeting varying standards in diverse markets (with as little as possible an overhead to the manufacturing process); and (5) being able to customize products to individual preferences. "Agility is dynamic and open-ended. There is no point at which a company or an organization has completed the journey to agility" (Goldman et al, 1995).

Furthermore, the capabilities outlined above require two contradictory attributes. On the one hand, an organization must mobilize enough resources to continually monitor the environment for changes in technologies, markets, competitors, political institutions which implies, as a first approximation, large organizations. On the other hand, agility is predicated on a certain nimbleness; at which large organizations are notoriously poor. A logical solution is for the firm to remain nimble by retaining only its core areas of competence and competitive advantage and supplement them with strategic and tactical alliances with other organizations with the requisite areas of competence. Hence, cooperation is central to agility; people and organizational culture must change to improve internal and external cooperation.

This might be achieved if the valuable role that cooperation plays in agility is appreciated and if it is understood to be non inconsistent with self-interest. Organizations must become flexible and distributed, with flattened managerial hierarchy, where decision-making authority is delegated and distributed. These changes support timely acquisition of information wherever it is generated, concurrent operations, and direct communication between participants, which in turn, lead to minimizing the cost and elapsed time of transmitting information.

People and organizational change alone are insufficient. They require technological assistance whose central contribution would be the acquisition, management, communication and reuse of information. Most companies turn to new technologies, in particular information systems (IS), that will provide them with a competitive edge or that will allow them to become agile. Unfortunately, technology alone is also insufficient. First, technology can be purchased by everybody, thus cannot be the underpinning of competitiveness. This applies to simple tools, advanced manufacturing methods, or enterprise integration tools. In relation to IS, it is the *content* of the tools—the enterprise knowledge and its continuous management—that could make the difference in competitiveness, rather than the tools themselves.

Second, the development of usable tools and moreover, the construction of usable models of enterprise knowledge, requires intimate understanding of the specific enterprise culture and its context which takes a long time and significant effort to acquire. Such intimate understanding is best formed through long-term relationships between customers and manufacturers of the tools allowing all participants to trace, follow, influence, and be influenced by the evolution of views of others. This implies that agility must be planned for and the capability to be agile must be evolved — *there is no agile way to prepare for agility.* Once the open-ended and dynamic nature of agility is understood, companies who have prepared for it would be able to exercises their flexibility on demand.

The third difficulty in relying on technology for becoming agile is related to the development of an agile design information system (ADIS). The substance of agile organizations involves forming virtual organizations or teams, each of which has its own established areas of expertise, cultural management practices, legacy tools, and historical records of previous designs with their, failures, successes, and modifications. Responding quickly to the creation of a virtual enterprise with the corresponding ADIS is extremely difficult considering the implications drawn in the previous paragraph. Furthermore, as the context of design (e.g., technology, markets, organizations, people) evolves continuously, an ADIS that supports agile design needs to be flexible enough to undergo constant evolution.

These difficulties add to the already complex nature of collaborative product design (CPD). In general, CPD's complexity results from its distributed nature with respect to many dimensions including: time, place, culture, practice, language, tools, expertise, discipline, perspective, etc. (Reich et al, 1996; Subrahmanian et al, 1993; Subrahmanian et al, 1997). Studies on computer-supported cooperative work (CSCW) have addressed some of this diversity but the systems that have emerged suffer mainly from (Reich et al, 1998): (1) inflexibility, (2) insufficient openness, and (3) unusability to both end-users and systems maintainers.

Given these difficulties and the stringent requirements of ADIS (Reich et al, 1998), it is unclear how an ADIS could be developed and what should be some of its attributes that would allow to develop it effectively. Over the course of building several information systems for different engineering organizations, we have developed or adapted responses to deal with these difficulties:

1. The insufficiency of technology is addressed by focusing on modeling the organization's knowledge into the ADIS and providing mechanisms to operate on the knowledge via these models. (When dealing with new virtual enterprises, the content is taken from the ADIS of the parent organizations.)

2. The need for intimate understanding of the organization is addressed by adopting participatory action research (PAR) or participatory design (PD) as the process philosophy (Reich et al, 1996).

3. The need to respond quickly to a variety of information needs is addressed by a flexible infrastructure for developing ADIS.

In this paper we review the *n*-dim system which embeds these responses, focusing mainly on the third aspect that deals with infrastructure flexibility. We discuss how *n*-dim's design and features support the development of ADIS from the perspective of IS developers and illustrate this support using several examples.

The remainder of this paper is organized as follows. Section 2 discusses the non-technical aspects that impact upon the success or failure of ADIS projects. Section 3 elaborates on the technical requirements of ADIS as emerging from the characteristics of CPD. Section 4 discusses the move from classical models of IS development to produce ADIS as the product of a virtual enterprise composed of, at least, the developers and the users of these systems. Section 5 describes the *n*-dim environment: an infrastructure for implementing ADIS. Section 6 presents examples of *n*-dim features that illustrate the flexibility of building applications and Section 7 summarizes the paper.

2 Non-technical aspects of ADIS development

There is growing evidence that information technology does not deliver its expected benefits (Ewusi-Menash, 1997; Gibbs, 1997; Strassmann, 1997). The fate of CSCW technology is no different. It has been common to develop increasingly more complex computer-supported systems and push them into the market without realizing their requirements in terms of cost and organizational change, and only paying attention to a more general evolutionary trend in human-computer relationships where users become more proficient in computer technology. However, early and even recent experiences with some CSCW technologies have not been successful (Lubich, 1995). Examples of failures include AT&T Picture Phone and some video-conferencing systems. Among the critical issues influencing these failures were non-technical including: the technology was not available to all the relevant community; it did not support the full intensity of interactions; it was not embedded into the normal work practice; it was not standard or interoperable with other tools; and it conflicted with social or organizational norms. These observations elaborate on the second difficulty mentioned in the introduction.

Part of these problems occurred due to the development process which did not include potential users in the teams. Increasingly, developers understand that any ADIS tool will have to match its context to work effectively (or to work at all), thus, must be designed as a collaborative effort much like other collaborative product developments. Even though these factors are known, failures continue to occur as was documented recently by the developers of the HI-TIME system (Brauer et al, 1996). These issues are been exacerbated when developing ADIS because by definition, design contexts are dynamic and evolve continuously.

In order to be agile, organizations need to acquire and sustain the five capabilities listed in the introduction. These capabilities can be attained only by carefully considering four key interacting aspects shown in Figures 1(a) and (b) (Lubich, 1995):

- the *people* involved in the collaborative product design through the extended enterprise;
- the *organization* (real or virtual) in which the people work;
- the *technology* used to support the collaboration; and
- the particular *task* that is the subject of the collaboration.

————————————

Put Figure 1 about here

————————————

Underlying the third difficulty mentioned in the introduction is another problem with existing CSCW tools: they assume fixed models of users or organizational cultures. They also assume a certain collaboration "worldview" such as information exchange, structural contingency, process losses, language action, workaday, etc. Such models and worldviews are then hard-wired into the CSCW system. There are three problems in this approach. First, in a context as diversified as CPD, and since groups and people are dynamic entities that evolve with time, no single worldview can be good for all times. People need the ability to articulate their activities, commitments, etc. and define their interaction mechanisms and negotiation policies. These cannot be determined *a priori* for all projects and situations. Furthermore, in the context of agile manufacturing, they are not known in advance. Second, users, organizations, their tasks, and the relative role of each factor change (Figure 1(c)). In fact, it is one of the critical goals of ADIS to create learning, adaptable, organizations. Therefore, fixed models will fail to evolve with their users. Third, experience shows that users often like to use systems in innovative, unexpected, yet beneficial ways. Systems that do not permit such uses will not be valued or used in the long run. Thus CSCW mechanisms need to have two important properties (Simone et al, 1995):

- *Linkability*: Since no single general notation for specifying work processes will suffice, different notations should be allowed to co-exist and features must be provided to facilitate their linking.
- *Malleability*: CSCW mechanisms need to facilitate making permanent and global changes to their behavior at the user and the system developer level.[1]

These two properties are interwoven in the characteristics of CPD discussed in the next section.

[1]The distinction between developer and maintainer is, in this case, vacuous.

3 Requirements of ADIS

The failures of some CSCW applications and the appreciation of the non-technical aspects of system implementation lead to re-examination of the requirements of ADIS systems. The re-examination results in 11 requirements which can be subdivided into two categories resulting from: (1) the distributed characteristics of CPD and (2) the evolutionary nature of the environment.

3.1 Characteristics of CPD

Diversity is a fundamental attribute of all CPD characteristics. In order to handle it, a system must be tolerant of diversity and support linkability. The following nine characteristics deal with technical and non-technical aspects of CPD.

1. Extended time. CPD activities extend over potentially long period of times. The context of design must be maintained over that period and longer, to address life-cycle issues and to allow for future reuse. The time dimension even manifests itself for single users working over an extended period of time (Monarch et al, 1997).

Support for this requires mechanisms for asynchronous communication, information sharing, and *history management.*

2. Multiple places. CPD activities take place in multiple locations. In the context of agile manufacturing, these locations may change over time.

Support for this requires communication facilities such as video-conferencing and whiteboards beside those that can support extended time collaboration. This characteristic introduces the issue of *availability* of computational resources and information which is critical for a distributed system. Additionally, asynchronous communication issues become important especially if the participants are scattered across significant space (and hence time zones).

3. Multiple cultures, practices, policies, and behaviors. Individual participants in CPD come from different cultures (e.g., egalitarian or capitalistic) whose impact on the adoption of CSCW technologies may be significant. Also, through their development, organizations develop distinct cultures consisting of different practices, policies, and behaviors.

Most tools are built with a set of basic mechanisms and embed certain worldviews. This may restrict the scope of these tools. In contrast, ADIS tools must accommodate this diversity and support the variety of phenomena associated with group interactions. In agile design, new teams are constantly formed whose make-up is diverse. In order to support work practices and policies, organizations participating in a virtual enterprise may be using different tools for coordinating work. When teaming for a joint project, the lack of a single management system may impair the ability to control the overall project. This necessitates the ability to *interoperate multiple* coordination systems. ADIS tools must be built such that this interoperation allows them to support all CPD requirements and also be customizable to a variety of group contexts. This customization must be relatively easy to allow end-users to control their environment rapidly.

4. Multiple languages. People from the same discipline but from different organizational departments or divisions often use different languages or terminologies to describe disciplinary knowledge (Sargent et al, 1992). People themselves also use different languages (unstructured/informal, e.g., text, images, audio, video; or structured/formal, e.g., equations, 3D models) to refer to different perspectives of the same objects (Subrahmanian et al, 1993).

ADIS tools must support this variety as well as be able to *translate* between languages as much as possible. When integrating legacy tools or extending existing systems, different programming languages are or must be used. This also needs to be supported.

5. Multiple tools. Some tasks such as word processing can be accomplished by different tools or methods. Different tools may be expected to be found in the same organization and certainly in different organizations that form the virtual enterprise. Moreover, existing organizations have significant investments in legacy tools that must be integrated into a new computational environment.

ADIS tools must support this multiplicity, potentially by an ability to translate between tools, or model the tools and their execution. In addition, ADIS tools must support easy integration of legacy systems. In order to alleviate the integration problem when new tools are introduced, preference should be given to tools that adhere to *accepted standards*.

6. Multiple expertise, discipline, or tasks. CPD engages people with multiple expertise in one discipline (vertical integration) as well as experts from multiple disciplines (horizontal integration).

ADIS tools must support the creation and use of such diverse expertise towards creating organizational shared memory (Konda et al, 1992). In addition, each task may involve multiple subtasks that need to be *coordinated* to achieve the overall task.

7. Multiple Perspective. People with the same expertise or from the same discipline may have different perspectives about a particular CPD if they assume different roles in the collaborative effort. In agile design, one person can sometimes act as a customer and in other cases as the developer or even be both simultaneously (e.g., someone in the middle of the supply chain). Perspectives evolve or are determined in response to the context of a particular project.

ADIS tools must support the integration of multiple perspectives from different disciplines and expertise. A minimal support for this can be achieved through the use of personal and group shared workspaces and the ability of participants to *tailor* their workspaces to their own perspective.

8. Interchangeable interaction methods. Interaction methods can be classified along a range from informing, coordinating, collaborating, to cooperating (Bair, 1989). A more refined analysis creates a 3-level notation that can cover many interaction mechanisms (Simone et al, 1995). People use all this variety of methods for communication.

ADIS tools must support anytime anyplace interaction methods in the same environment with the ability to switch back and forth between them. The development and evolution of interaction metaphors need to be supported for these interaction types (e.g., bulletin boards are appropriate for asynchronous exchange of messages). Note that in agile design communication involves the complete value-added chain from manufacturers to customers. These groups may have completely different interaction practices.

9. Usability and adaptability to workers with different levels of education (including computer-literacy). By and large, most CSCW tools described in the literature are developed for collaboration of experts familiar with the use of computers, but more importantly, by experts in computers that may not appreciate the difficulties that regular users may have. Even when users are computer experts, applications could fail due to usability deficiencies. In agile design, no assumption about the proficiency of design participants (customers as well as designers) with computers can be made.

Therefore, in the building of ADIS, *usability* for diverse end-users is a prime issue. Participatory design (PD) (Reich et al, 1996) and system interface flexibility can assist to address it.

3.2 Evolutionary nature of agile design

No characteristic of agile design remains constant over time. This nature is depicted in Figure 1(c). The task, organizational structure, people, and technology, as well as their relations and relative contribution to a project evolve. Thus, it is wrong to assume that employing a PD approach to system development is sufficient to guarantee the system's success for an extended period of time.

10. Group evolution and learning. Organization evolve with time. In agile manufacturing, the composition of teams may change, their commitments may change, they may change their tasks or task definitions, team boundaries change as organizational relationships are formed, and organizational cultures evolve.

It is critical that whatever change occurs, a historical record of the progression is maintained for future reference. ADIS tools must support this *evolution, dynamism*, and history management.

11. Supporting variable-term relationships. Participating in agile enterprises entails having variable-term relationships with other organizations or individuals. The need to cooperate mandates wide communication channels and *access* to information, yet it also requires that *security* of information not be compromised. Variable term relationships also create issues of *reliability*, *quality*, and *availability* of services or information. Finally, in order to support a desired mode of long-term cooperation, the *history* of a design product and the cooperation activity needs to be accumulated and organized.

4 Moving from production of ADIS to agile production

People collaborate by (1) manually creating, organizing, maintaining, visualizing, and reusing information in various media and modes, (2) manipulating information by various tools, and (3) sharing and communicating information (Subrahmanian et al, 1997; Reich et al, 1998). We will refer to these as *information management activities (IMA).* Computational tools for supporting cooperative work of any form must delineate how they support IMA related to the task of interest. In the context of CPD or agile design, the aforementioned eleven characteristics manifest and must be supported in the design of ADIS. This section discusses existing work on CSCW and ADIS development and ends with an approach that addresses the full spectrum of CPD characteristics.

4.1 Existing approaches to CSCW and ADIS development

It has been common for CSCW studies to focus on the interaction mechanisms, coordination, scheduling, and workflow, of people in existing organizations. By and large, the product of the "work" studied in CSCW has been documents. These systems have not aspired to provide support for *all* tasks performed towards completing design. Collaborative CAD tools deal with similar issues while the product of the work is CAD models. Figure 2 shows these two technical areas with their activity focus, basic core models, and examples of commercial tools. Clearly, any work practice span a range of activities from informal to formal (Subrahmanian et al, 1993) and consists of vertical (disciplinary) and horizontal (interdisciplinary) activities (Konda et al, 1992). In this context, we view ADIS as integration systems that support the full range of activities in a flexible manner, by integrating facilities provided by different tools ot components.

————————————

Put Figure 2 about here

————————————

The limitations of many early CSCW tools prompted researchers to build toolkits for configuring CSCW or collaborative CAD systems thus introducing some flexibility into these tools. The philosophy of these toolkits may differ with one another but they all attempt to include building blocks that can provide greater flexibility in matching a support system to a particular collaborative work context. For example, EGRET (Johnson, 1994) is a research CSCW toolkit based on multi-user hypertext model with typed data models that supports structural evolution of schema by users. It collects significant state information where state includes information about users, artifacts, the context of collaboration and other information. This information can be used for replaying history, revision control, automated software configuration through machine learning, and tool invocation (Johnson, 1996). Unfortunately, the collection of this state burdens applications built with EGRET making it suitable only for "collaboration-in-the-small".

Lotus Notes is a hypertext, multimedia, document-based, collaboration information system. Collaboration is achieved through document publishing and sharing. Notes acts as a central access resource to electronic mail, databases, WWW, desktop tools, etc. Version control can track changes made by different people. Collaborative activities can be coordinated via notification mechanisms and group calendaring and scheduling. Asynchronous communication is facilitated by electronic mail and synchronous communication is supported by video-conferencing. Security is provided at four levels: authentication, access control lists, field-level privacy through encryption, and digital signatures. Lotus Notes provides facilities for configuring CSCW systems for particular needs and also provides an object-oriented (OO) scripting language for building complex applications. Finally, Notes provides an API that allows it to connect to external applications.

Cælum is a team design support system developed by Toyota. It is built around a central CAD model with various integrated services and functions for developing applications. The environment allows for customizing the user interface, adding new commands as macros, and developing programs and linking them to the environment. The system embeds its own PDM system but can be linked to other products as well. Basic CSCW facilities such as electronic mail and video-conferencing are provided and mechanisms are available for tracing incremental changes in designs. Cælum has three levels of customization and extension to the basic environment, the latter allowing arbitrary programs to be linked.

EGRET, Notes, and Cælum address critical needs of computational support. However, they are insufficient to deal with the full complexity of agile design. Even if different organizations employ the same toolkit, they would still have different specific implementations of various mechanisms. Thus, different systems might have to be made interoperable or new ones put into place quickly (see requirement #3).

Many other studies dealing with computer support for CPD have focused on creating networks of services over the Internet, with communication protocols for data transfer at various levels such as FTP or MIME for data in general or IGES or PDES/STEP for graphic or product models, respectively. Recently, studies have begun to adopt HTTP and/or CORBA protocols for transferring multimedia and object information over the network, and moreover, for locating and using services over the Internet (Erkes et al, 1996; Park et al, 1993).

However, the focus of these studies has been on the *mechanics* of the communication or interaction. The issues that dealt with the meaning of the work were expected to be resolved, or avoided, by the introduction of various standards such as standard trade agreements, standard forms for communication, standard libraries of part descriptions, standard ways of accessing services or cooperation, etc.

The issues of quality or reliability of services are also expected to be resolved by standard pre-qualification. The outcome of this approach will be short-term “cooperation”, rigid (standard) business processes, and little chance to benefit from long-term relationships that are critical to cost reduction and quality improvement (Goldman et al, 1995). Furthermore, the effort required for setting such inclusive, *a priori* standards might take much too long before it becomes even partially practical. Another limitation of this approach is that in the initial design stages, partners negotiate items such as problem understanding, shared terminology, design issues, and the cooperation strategy. Constraining them into standard ways of doing things might be unacceptable.

This criticism does not diminish the value of standards; they can still serve as baselines for discussions and agreements. Nevertheless, the rule, and *one that directly follows one premise of agile practice*, is that producers of ADIS should be agile manufacturers by learning to provide solutions (including development and maintenance) for other agile manufacturers, rather than, standardized products.

4.2 Agile production of ADIS

The key premise of agility is that virtual organizations could be created quickly to address dynamic manufacturing needs. These organizations might need to re-organize quickly or adopt/adapt new technology. ADIS tools must be responsive to the quick integration and modification of the IS to address these needs. This requirement is related to the evolutionary requirements from Section 3.2, but even more stringent since such integration may involve quick rather than evolutionary modifications to software. The diversity and evolutionary properties of CPD must be supported by flexible computational mechanisms that could be manipulated by end-users and developers. The responsibility of evolving the system depends on the particular modification required.

In order to support agile manufacturing by building information systems, these systems must be developed quickly as the product of agile design. Figure 3(a) illustrates the move from traditional to agile manufacturing as discussed in the introduction. Being an agile manufacturer of ADIS entails adopting a particular view of system development. Figure 3(b) shows the specialization of (a) to software development. Instead of the traditional waterfall software development process (Boehm, 1988), we now have an evolutionary process guided by a methodological emphasis on collaborative prototyping or participatory design (Budde et al, 1992; Reich et al, 1996). Moreover, the relations between developers and users extend over time and are characterized by intimate cooperation. This approach can vastly benefit from the support of special toolkits. Hence, this approach differs from traditional approaches in that it accommodates our understanding that the design, implementation, and deployment of ADIS is itself an agile design process. Hence, beside supporting the development of a system that will support CPD, our approach emphasizes the importance of enabling the quick, flexible development of such systems. This support is translated into a particular philosophy and approach that is elaborated on in the next section.

————————————

Put Figure 3 about here

————————————

5 *n*-dim: A system for evolving agile design support systems

n-dim (n dimensional information modeling) (Levy et al, 1993; Subrahmanian et al, 1991) is a flexible computational environment built to *facilitate* and *study* collaborative design from the initiation of a design process and continuing throughout the life-cycle of the artifact. This section briefly discusses our philosophy of using *n*-dim, its basic technical description and motivates its usefulness as a tool for building ADIS.

5.1 Philosophy of using *n*-dim

n-dim embodies a philosophy and implementation methods for building specific applications within a given environment (see (I) in Figure 4). At the foundation (1), there is a software infrastructure designed to address the list of requirements presented in Section 3 and also designed to scale up to handle real applications. As additional applications are developed, *n*-dim would include repositories of various blocks for building applications (2). At the top level (3), our research and development follows philosophical positions and theories we developed and evolved through empirical studies (Dutoit, 1996; Finger et al, 1993; Konda et al, 1992; Monarch et al, 1997; Reddy, 1996; Reich et al, 1996; Sargent et al, 1992; Subrahmanian et al, 1993; Wilkins et al, 1989). These theories guide us in future studies and development projects and are the subject to constant reflection (4).

A project starts as a collaboration with industrial or other partner(s). In order to support design and study it at the same time, we adopt participatory action research (PAR) as our development methodology (Reich et al, 1996). Together with our collaborators, we define project goals (5). The development process (6) uses the infrastructure and reuses the repositories of previous blocks (7) for prototyping the application (8). The development, in turn, enriches the repositories and the infrastructure. This process iterates until the goals, as understood at each iteration, are satisfied by the evolving application (9). The collaborative project is studied and reflected upon continuously (10). Its results are used to refine our theories (11).

In order to study design as it evolves, we use the explicit features of *n*-dim such as history capture as well as other, implicit state information we can collect using the built-in instrumentation in *n*-dim to capture a variety of event histories (down to, in principle, individual key strokes).

————————————

Put Figure 4 about here

5.2 Technical details of *n*-dim

This subsection describes several aspects of *n*-dim basic core model: generalized-graphs.

Generalized modeling over a flat space of objects.

The notion of *flat space* captures the fact that an individual object can be situated in multiple contexts. The space of objects in *n*-dim is conceptually flat. Multiple structures can be imposed on this flat space by means of *models* which are directed hyper-graphs. Figure 8 shows one such model.

Generalized modeling allows people to operate on things and kinds of things interchangeably, and move freely from one level of abstraction to the other. All but published models (see below) are mutable, and can have operations and attributes defined on them without affecting any other model. This means that models (information structures) can serve as *prototypes:* potential progenitors of other models. Any model can serve as the starting point for another model, in the sense that it can be copied and the copy modified in ways independent of the original (though not via the class/instance relationship). In a preliminary design phase the evolution of prototypes can be free-wheeling, but nevertheless constrained and manageable. Exploration of ideas can be structured without having to commit to a taxonomic structure that prevents consideration of important ramifications.

Generalized modeling over a flat space allows using different terminologies (by naming the same object in different models differently) and supports multiple perspectives of information.

Models can also serve as *modeling languages*; that is, as templates for structuring information. Modeling languages themselves are represented as models (i.e., as things to be designed, negotiated, etc.). In this sense, models can act as types for the creation of other models. Modeling languages are codifications of inter-relationships between information objects that are found useful throughout and beyond a current design situation. These languages can be used to codify formal and informal knowledge (Subrahmanian et al, 1993). For example, Figure 8 displays a model of the interface to part of an application called the Developer Queue Viewer (see Figure 7). The model is written in the GUI (graphical user interface) language. This language allows to model the structure of the interface from given building blocks.

Languages for structuring formal knowledge can be used to code expressions, constraints, production rules, and other graph-based models such as conceptual structures. Through such conceptual structures, *n*-dim could interface knowledge expressed in KIF format (Sowa, 1993) or use product model information represented in EXPRESS (Wermelinger and Bejan, 1993). Informal languages can be used to annotate documents, drawings, and formal structures of knowledge. These languages provide the necessary facility for coding design knowledge, organizational procedures and tools, expertise, etc. The specification of a design can be increasingly standardized (but evolving from and tailored to the particular enterprise design practice) and automated.

Within a single project, design history can be captured as a sequence of loosely constrained prototypes finally evolving into a modeling language that is the progenitor of a more tightly constrained and less varying sequence of prototypes, as, for example, in the designs of a mature product. Figure 10(b) shows a pedigree model constructed automatically that shows the evolution of one model into another. Such a model can be used to trace the history of models. Design histories can also be charted and organized for continued elaboration and refinement across projects, organizational entities, and cultural contexts.

Following the library as a metaphor, *publishing* is a mechanism for making models formally exchangeable and persistent. Hence, *traces* of the evolution of information and its structure over time can be found in the growing repository of published objects. Design history in *n*-dim is treated differently than it is in traditional revision control systems. History is viewed here as a connected sequence of models created by designers. The maintenance of history is achieved by the publishing mechanism. Once a user publishes a model it is immediately available to anybody in the group or organization subject to access controls as a means to control knowledge sharing. A published model is unalterable (persistent) but is copyable by any person in the group. Thus, *n*-dim both facilitates the act of collaboration and captures design history as a by-product. In short, design history is created and maintained as part of the process of creating designs.

Communication facilities including integration of multi-media information.

Publishing and *searching* are critical activities in modeling by multiple design participants. They provide the basic substance for *asynchronous communication.* Agile design can easily generate an enormous amount of information in the form of published models. As the corpus of information increases continually, facilities are needed to search it for information relevant to current modeling activities. Currently, search is performed via a structured query editor. *n*-dim allows for, and will include, techniques designed to facilitate search by models, whereby a user could specify a partial model and search for models closely related to the partial model.

n-dim's modeling facilities also allow for capturing other forms of context such as drawings, photographs, audio, and video. We intend to consider how to include video and remote multimedia interaction as part of our distributed collaboration improvement efforts. This integration task would entail the collection of passive, off-line multimedia recording (meetings, demonstrations, background presentations, etc.) and the development and evaluation of an integrated, on-line recording, teleconferencing, archiving, and retrieval facility. Each of these activities is included in our effort to develop and implement collaboration support systems.

Integration of CAD/CAM and legacy tools.

The n-dim architecture actively aids in the integration of information from CAE tools. For example, a layout from a particular CAD tool can be related to appropriate information such as design decisions, test data, etc. The level of integration includes (1) black-boxes where n-dim only allows for the management of inputs into and outputs from the tools in context; (2) the provision of mechanisms for aiding tool integration through neutral product models such as PDES/STEP; these latter can themselves be represented as n-dim models; to (3) the encapsulation of code written in diverse languages such as C or Fortran. Thus, enterprise legacy tools can be linked to n-dim and treated like other n-dim objects. The n-dim environment can also be configured and extended using the native language used to implement it (i.e., BOS/stitch (Dutoit et al, 1996)).

5.3 Summary

Figure 5 illustrates how the different n-dim technical features help address the 11 special characteristics of CPD. A "+" in the figure denotes a strong positive impact of the feature (on the column) on the characteristic (on the row). For example, The flat space feature supports the use of multiple terminologies and perspectives; also, repositories improve the usability of n-dim and the ease of building applications. Although some features are prerequisites to others (e.g., the publication feature is necessary to address the history keeping feature, the features are almost independent of each other. This suggests that we have uncovered a small, if not minimal, set of features needed to address the complex requirements of CPD (Subrahmanian et al, 1997).

Figure 5 can be elaborated by introducing the dependencies between the characteristics themselves and between n-dim's features. For example, one has to deal with multiple languages before the organizational cultural diversity requirement can be addressed. Upon introducing these inter-relations, the figure quickly resembles the House of Quality, a central QFD (quality function deployment) tool for designing a system starting from its general requirements. We are actually using QFD to design the next version of n-dim.

Put Figure 5 about here

5.4 Scaling-up n-dim

n-dim is designed to be portable, with a layered architecture that should scale up to deal with the intensive storage and other computational demands of real engineering tasks. The layered architecture allows to replace complete layers (such as the user interface or the database) with new or other layers as deemed necessary for particular projects or for improving the basic infrastructure. This allows to adopt a legacy interface or database that our collaborators use. To illustrate, Figure 6 shows one way of turning n-dim into a server accessed through common web browsers over a network by replacing its native user interface with an HTTP front end.

————————————

Put Figure 6 about here

————————————

n-dim is also being designed to handle very large applications. To help achieve this, objects themselves (especially large objects) need not be stored in the database; instead, only the attributes necessary to search for such objects (e.g., the intrinsic attributes) need to be stored internally. The objects themselves can be stored locally on individual workstations, local servers, distributed or remote servers. This separation between the space in which objects are stored vs. the space in which attributes about them are kept for the purpose of search can be used to great advantage in scaling *n*-dim for large (hundreds of thousands to millions of objects) applications.

Finally, *n*-dim is being designed to handle many organizational structures via configurations based on what in *n*-dim is called a *cell*. The collection of a set of database processes and workspace processes is called a cell. A broad range of cell configurations is possible, from a single, centralized database with several users clustered about it (a small work group), to a totally distributed configuration with both central and localized components for individual users (entire organization or sub-organization, or clusters of smaller work groups).

6 Experience in developing tools with *n*-dim

The following examples briefly illustrate some of the features and applications of *n*-dim we mentioned previously and that are directly related to agility. These examples are only meant to illustrate ideas and are not an exposition of *n*-dim range and depth of capabilities.

6.1 Flexibility in creating communication facilities

Figure 7 shows a simple application we are using to manage *n*-dim's bug reports. The application allows to access the description of bugs, correct it, create references to any *n*-dim model which can be additional material including the code itself, or assign the bug to another developer. This tool is a collaborative tool between users, developers and among the developers themselves.

————————————

Put Figure 7 about here

————————————

The model of the application in *n*-dim is shown in Figure 8. The model is written in a modeling language that allows to have objects of type Widgets and required links in models. In this model, the Developer Queue Viewer object is linked to ML Listbox and Bug Viewer by the required link. This object contains information about how to assemble the required components into the application and the required links tell where to look for the information needed. The required links show that MO Listbox appears 3 times in the application, once in each of the objects requiring it (i.e., Bugs, Bug Reports, and References). In such a way, a developer can model or “program” large applications from existing building blocks.

————————————

Put Figure 8 about here

————————————

Another class of collaboration applications we developed includes several issue-based discussion tools. We have used them in several situations (Coyne et al, 1994). A simple example of such a tool is shown in Figure 9. This application was modeled as shown in Figure 10(c). The listbox object was reused from the Bugs application (Figure 10(a)), however, it had to be modified to include another operation. This modification was recorded automatically by a pedigree model (Figure 10(b)) as a history keeping mechanism. The modification was simple and the reuse immediate.

————————————

Put Figures 9 and 10 about here

————————————

6.2 Incorporation of tools

We have integrated tools with *n*-dim to varying degrees. In the ACORN project, we have used *n*-dim to capture the trace of interactions with specific engineering design services provided by distributed organizations on the Internet specifically using the WWW technology. More particularly, the goal was to provide a history recording mechanism which could serve at least two purposes. First, to maintain a record of the exchanges between the engineer and a service much as one would have copies of paper-based exchanges. Second, to use the record to generate a template process for the future use of either this particular service or other WWW-based engineering services. Additionally, if the history could be enriched with experience reports, annotations, etc. from both the engineer and others involved in the process (such as the purchasing department, the quality control department, etc.) the trace would be much richer in terms of future re-use of both the historical record as well as the service itself. Finally, since no engineering activity is done in isolation, the selected history capture mechanism must be extensible to include other groups such as management, purchasing, manufacturing, quality control, testing, etc.

Our approach was to let the engineer use any browser of choice and let *n*-dim be the history and rationale capture system, along with one of several WWW servers capable of providing proxy services. A proxy server, in this case, is a server that receives all URL requests from the browser irrespective of the URL. We have also developed other Internet-based applications with *n*-dim whose architecture is shown in Figure 6 (Konda et al, 1997). Using *n*-dim, an organization could capture the interactions with its service providers; allow its collaborators some degree of access to its information system; or integrate with its partners seamlessly if they also use *n*-dim.

We have loosely integrated other systems into *n*-dim and also performed tight integrations of external tools. For example, we integrated a system for Asea Brown Boveri where *n*-dim was used as a history and rationale capture tool. We have also integrated the ASCEND equation-based modeling environment with *n*-dim (Thomas, 1996). *n*-dim was used to capture modeling sessions including: models used, their revisions, and their underlying rationale and the results of the modeling sessions. Subsequently, *n*-dim allows users to reuse effectively previous modeling scenarios.

These examples demonstrated some of *n*-dim's key concepts: generalized modeling using a graph-based environment, quickly creating communication facilities, the ease of building applications by modeling, integration of external legacy tools at various levels, and the capture of product and process history.

6.3 Creating repositories or libraries

Flexibility and generality could be the biggest enemies of successful implementation. In order to circumvent potential inefficiencies, we must promote reusability and modifiability or previously successfully used components. To achieve this, we are working on implementing generic repository management mechanisms in *n*-dim that will be used to implement repositories of various building blocks such as modeling languages, concepts, operators, etc. These repositories will allow us to improve the effectiveness of reuse at the granularity described. Note that our reuse is very different from the common notion of reuse as discussed, for example, in OO languages. There, one may be able to reuse class hierarchies. In our use of a prototype-based language (Dutoit et al, 1996), we have a much greater flexibility: we can reuse instantiated objects and even mix parts of objects or operators for even more effective reuse.

In the AEC project (sponsored by a European company) we have been developing a document repository that stores project experiences. We improved upon this repository in the CODES project whose goal is to understand the process of putting together a design system for collaboration among designers in an integrated computational environment for managing information, tools and design process history. This objective is demonstrated by creating assembly based design specific environment using *n*-dim where tools using different models of integration will co-exist along with repositories of product information and design experiences. All these could be used in a collaborative design setting. One outcome of the project is the ability to create composable collaborative design systems from repositories of building blocks.

6.4 Advantages of *n*-dim

n-dim offers numerous advantages compared to other tools or approaches to develop ADIS.

Comprehensiveness. *n*-dim is a comprehensive approach to dealing with software development and knowledge development in an evolutionary manner.

Uniformity. The system combines evolution, history, and modeling within the same framework - the framework of graph-based modeling.

Modeling Sufficiency. Graph-based modeling was found sufficient to support all the range from intuitive or informal to formal modeling we encountered in our projects.

Integration. *n*-dim facilities were capable of supporting the easy integration of legacy tools at various levels (Dutoit et al, 1996). *n*-dim also integrates well with present trends in computing: WWW, client/server architecture, distributed databases, etc.

Modularity. *n*-dim layered architecture supports replacing complete layers with legacy layers or others more suitable to a particular project with minimal effort (Dutoit et al, 1996). For example, we distribute a beta-version of *n*-dim with a public domain database instead of the commercial one we use for developing systems, and without changing the other parts of *n*-dim.

History capture. In all our projects we demonstrated the capability of *n*-dim to serve as a history capture and replay mechanism for various design activities up to the level of keystrokes, and to serve as a growing repository of organizational knowledge.

Customization. In principle, *n*-dim is developed to support the easy customization of applications. Currently, part of the customization that can be accomplished using graph-based modeling is easy but other parts, while supported, require the use of a programming language we developed (Dutoit et al, 1996). Fully supporting customization at the users level is one of our primary infrastructure development efforts.

Task coverage. *n*-dim provides a full coverage of modeling activities from conceptual to life-cycle issues. It is different from, e.g., PDM or workflow systems because it can be used by all organization members in doing all their work and not just be a coordination/distributer of work.

Evolutionary. *n*-dim supports easy, natural, and incremental shift from paper to computer-based engineering information management.

Project resources. It is relatively quick to develop *n*-dim applications and deploy them. An organization can start with a pilot system, learn the technology, assimilate it, and then decide how to proceed.

Understanding. Our approach allows company managers to better appreciate how their company really operates (which may not necessarily follow the written policies). Some executives view this as one of the important ingredient of the project.

Best practices. Large organizations composed of a large number of divisions or smaller companies could evolve a set of best practices and share them. We do not have experience in this aspect but can offer an explanation how this is supported by our approach.

Scalability. Although n-dim grew out of a research project, our goals mandated that it can scale up to deal with real applications - a capability most research tools do not have.

Requisite variety. A system must have the "right" ingredients in the "right" granularity, so that they could be assembled to address the variety in design contexts. Discovering a minimal set of ingredients allows to build complex systems with minimal effort spent on infrastructure development. Our experience shows that n-dim's ingredients are sufficient to address most issues.

6.5 Observations or apparent limitations of n-dim

Our projects and discussions with researchers and practitioners have led us to several observations or apparent limitations of n-dim. We believe that most other CSCW or ADIS tools exhibit similar properties.

Initial effort. A pilot application with an organization takes more time than latter applications. The reasons include: the need to establish project work practices, teach the technology, learn about company practices and tools, etc. Nevertheless, we think that this apparent limitation manifests itself worse in relation to other tools.

Customer responsiveness. Whereas top managers understand our approach very well, especially in Europe and Japan, lower level practitioners are sometimes harder to convince to participate in projects. Nevertheless, we think that our situation is better than those trying to promote other approaches to building ADIS.

Cultural acceptability. Our experience suggests that we have more openness to our approach in Europe or Japan than in the US.

Infrastructure maintenance. We discovered that we need to spend time on maintaining our infrastructure in order to maintain and improve our strength to prototype quickly.

Expertise overlap. In order to operate well, the n-dim group needs a certain overlap in expertise and experience. We think that this requirement exists in relation to other organizations trying to be agile.

6.6 Barriers for using computer tools

The following barriers are not particular to n-dim but relate to all computational technologies.

Computerized. An organization must have sufficient computer and networking infrastructure and be willing to adopt this technology for conducting work, even if in an initial small scale.

Change. In order to succeed in a project, an organization must be willing to accept cultural change (although only incremental).

Commitment. We found that project success requires strong top-management commitment and participation with wide-communication bandwidth with potential users. Our present project (i.e., AEC) has such support and we see that project quality improves due to this (as well as due to other reasons).

Participation. We have found that a major hurdle in the implementation of support systems was the lack of involvement of end users in the process. It was insufficient to involve research groups of product divisions in order to assure acceptability.

7 Summary

Starting from reviewing what agility means and that there is no agile way to achieve it, we briefly described the dimensions of agility and how they can be attained by organizational means. These means must be supported by an information integration infrastructure.

We discussed the complexity of CPD, its evolutionary nature and the difficulty it endangered on the development of ADIS. Common methods of software development cannot cope with these difficulties thus may impede the responsiveness to information management needs of a planned virtual organization.

We cannot be agile as developers of ADIS unless we prepare our methods and infrastructure. Our approach has been to develop methods, a flexible infrastructure called *n*-dim and repositories of different building blocks from which we can effectively develop new applications. Our methods allow us to quickly understand organizations' information management needs, and as a complement, our tools allow us to quickly prototype ADIS and hence, support the agility of our collaborators.

As agile developers of ADIS, we seek to constantly learn about the effectiveness of our development philosophy, methods, and tools. We design this learning capability into our tools by capturing history that can subsequently be studied, by closely collaborating with organizations, and by developing an infrastructure that supports quick prototyping. The layered architecture allows us to quickly incorporate new technology that can improve *n*-dim's functionality. The integrative capability supports the incorporation of new functionalities, facilities, or services that prove useful. As a research group, we have limited resources to demonstrate our agility as defined by the five capabilities in the introduction, nevertheless, we believe that the approach we presented is a solid foundation of such capabilities.

References

Bair, J. H. (1989). "Supporting cooperative work with computers: Addressing meeting mania." In *Digest of Papers. COMPCON Spring '89. Thirty-Fourth IEEE Computer*

Society International Conference: Intellectual Leverage, pages 208–217, New York, NY, IEEE Computer Society Press.

Boehm, B. W. (1988). "A spiral model of software development and enhancement." *IEEE Computer*, May:61–72.

Brauer, D., Johnson, P. M., and Moore, C. (1996). "Requiem for the project HI-TIME collaborative process." Technical Report ICS-TR-96-04, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii.

Budde, R., Kautz, K., Kuhlenkamp, K., and Zullighoven, H. (1992). *Prototyping: An Approach To Evolutionary System Development*, Springer-Verlag, Berlin.

Coyne, R., Dutoit, A., Uzmack, J., and O'Toole, K. (1994). "IWEB (Information WEB): Information management for software." Technical Report EDRC-05-87-94, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Dutoit, A. H., Levy, S. N., Cunningham, D., and Patrick, R. (1996). "The Basic Object System: Supporting a spectrum from prototypes to hardened code." In *Proceedings of OOPSLA '96*, pages 104–121, New York, NY, ACM.

Dutoit, A. (1996). "The role of communication in team-based software engineering projects." PhD thesis, Department of Electrical Engineering, Carnegie Mellon University, Pittsburgh, PA.

Erkes, J. W., Kenny, K. B., Lewis, J. W., Sarachan, B. D., Soboiewski, M. W., and Sun, J. R. N. (1996). "Implementing shared manufacturing services on the world-wide web." *Communications of the ACM*, 39(2):34–45.

Ewusi-Menash, K. (1997). "Critical issues in abandoned information systems development projects." *Communications of the ACM*, 40(9):75–80.

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.

Gibbs, W. W. (1997). "Taking computers to task." *Scientific American*, July:82–89.

Goldman, S. L., Nagel, R. N., and Preiss, K. (1995). *Agile competitors and virtual organizations*, Van Nosstrand Reinhold, New York, NY.

Johnson, P. M. (1994). "Experiences with EGRET: An exploratory group work environment." *Collaborative Computing*, 1(1):87–107.

Johnson, P. M. (1996). "State as an organizing principle for CSCW architectures." Technical Report ICS-TR-96-05, Department of Information and Computer Sciences, University of Hawaii, Honolulu, Hawaii.

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.

Konda, S. L., Reich, Y., Subrahmanian, E., Terk, M., Cunningham, D., Patrick, R., Thomas, M., Westerberg, A. W., and Dutoit, A. (1997). "Networked information systems for collaborative product development in virtual enterprises." In *(submitted for publication)*.

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.

Lubich, H. P. (1995). *Towards a CSCW Framework for Scientific Cooperation in Europe*, Springer-Verlag, Berlin.

Monarch, I. A., Konda, S. L., Levy, S. N., Reich, Y., Subrahmanian, E., and Ulrich, C. (1997). "Mapping sociotechnical networks in the making." In Bowker, G. C., Star, L. S., Turner, W., and Gasser, L., editors, *Social Science, Technical Systems, and Cooperative Work*, pages 331–354, Hillsdale, NJ, Lawrence Erlbaum.

Park, H., Tenenbaum, J. M., and Dove, R. (1993). "Agile infrastructure for manufacturing systems (AIMS)." In *Proceedings of DMC'93*.

Reddy, J. C. (1996). "Design as artifact theory building." PhD thesis, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA.

Reich, Y., Konda, S. L., Levy, S. N., Monarch, I. A., and Subrahmanian, E. (1996). "Varieties and issues of participation and design." *Design Studies*, 17(2):165–180.

Reich, Y., Konda, S. L., and Subrahmanian, E. (1998). "Requirements of information systems for collaborative product development.". submitted.

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.

Simone, C., Divitini, M., and Schmidt, K. (1995). "A notation for malleable and interable coordination mechanisms for CSCW systems." In Ellis, C., editor, *Proceedings of the Conference on Organizational Computing Systems. COOCS '95*, pages 44–54, New York, NY, ACM Press.

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.

Strassmann, P. A. (1997). *The Squandered Computer*, The Information Economics Press.

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.

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., Reich, Y., Konda, S. L., Dutoit, A., Cunningham, D., Patrick, R., Thomas, M., and Westerberg, A. W. (1997). "The *n*-dim approach to building design support systems." In *Proceedings of ASME Design Theory and Methodology DTM '97*, New York, NY, ASME.

Thomas, M. E. (1996). "Tool and information management in engineering design." PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA. Available as EDRC Technical Report EDRC 02-38-96.

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.

Wilkins, D. J., Henshaw, J. M., Munson-Mcgee, S. H., Solberg, J. J., Heim, J. A., Moore, J., Westerberg, A., Subrahmanian, E., Gursoz, L., Miller, R. A., and Glozer, G. (1989). "CINERG: A design discovery experiment." In *NSF Engineering Research Conference*, pages 161–182, Amherst, MA, College of Engineering, University of Massachusetts.

8 Acronyms

ADIS:	Agile Design Information System
API:	Application Programming Interface
CAD:	Computer Aided Design
CAE:	Computer Aided Engineering
CPD:	Collaborative Product Design
CORBA:	Common Object Resource Broker Architecture
CSCW:	Computer-Supported Cooperative Work
FTP:	File Transfer Protocol
GUI:	Graphical User Interface
HTML:	Hypertext Markup Language

HTTP: Hypertext Transfer Protocol
KIF: Knowledge Interchange Format
IGES: Initial Graphics Exchange Specification
IMA: Information Management Activities

MIME: Multipurpose Internet Mail Extensions
OO: Object Oriented
PAR: Participatory Action Research
PD: Participatory Design
PDES: Product Data Exchange using STEP

PDM: Product Data Management
QFD: Quality Function Deployment
STEP: Standard for the Exchange of Product Model Data
URL: Uniform Resource Locator
WWW: World Wide Web

Figure Captions:

1. Issues affecting CSCW implementation [(a) and (b) adapted from (Lubich, 1995)]
2. ADIS: Spanning a range including CSCW and Collaborative CAD tools
3. Agile development of software for agile design
4. The *n*-dim project philosophy: iterative study and development
5. Influence of *n*-dim's features on attaining ADIS requirements
6. Creating an *n*-dim server for Internet-based collaboration
7. The Bug Queue Viewer application
8. The *n*-dim model of the Bug Queue Viewer application
9. The Issue-based discussion application
10. The *n*-dim model of the Issue-based discussion application

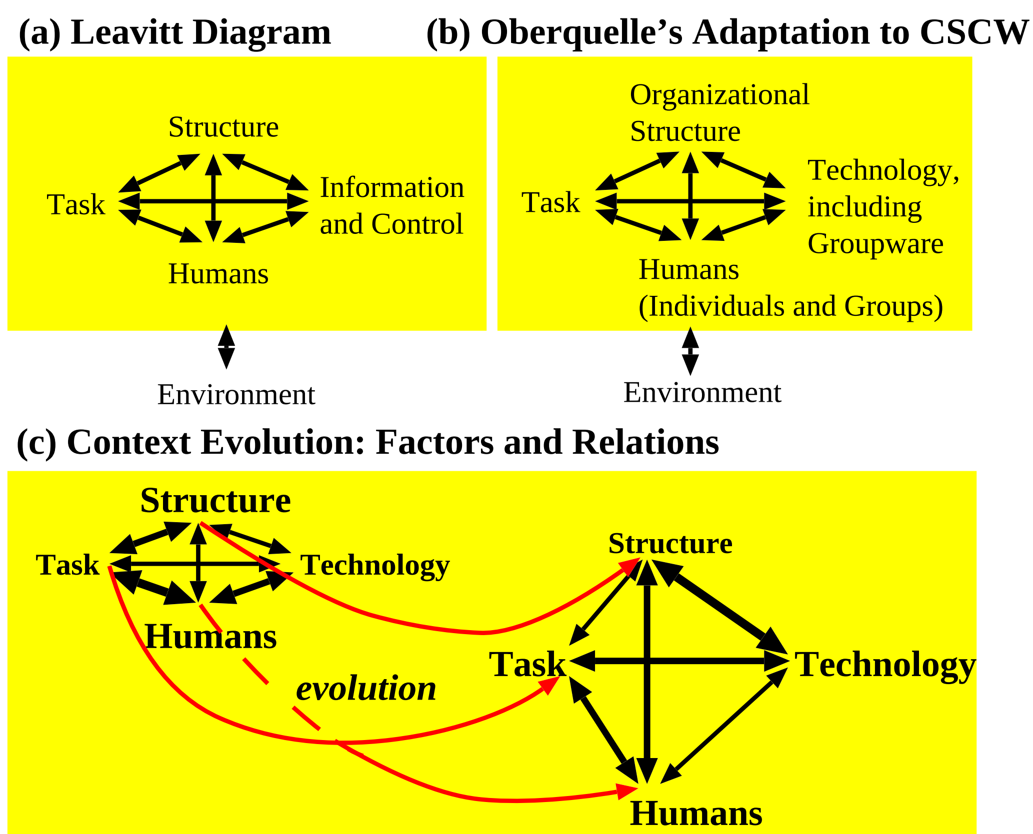

Figure 1: Issues affecting CSCW implementation [(a) and (b) adapted from (Lubich, 1995)]

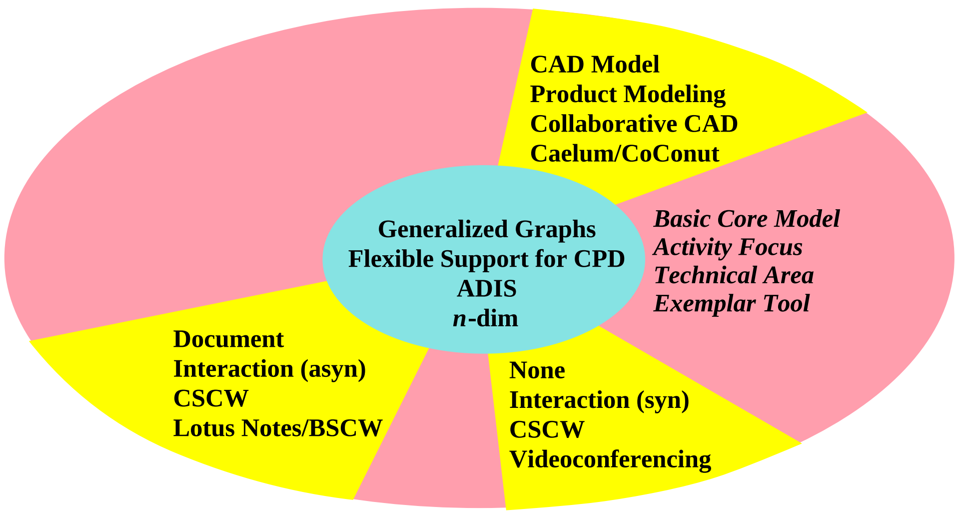

Figure 2: ADIS: Spanning a range including CSCW and Collaborative CAD tool

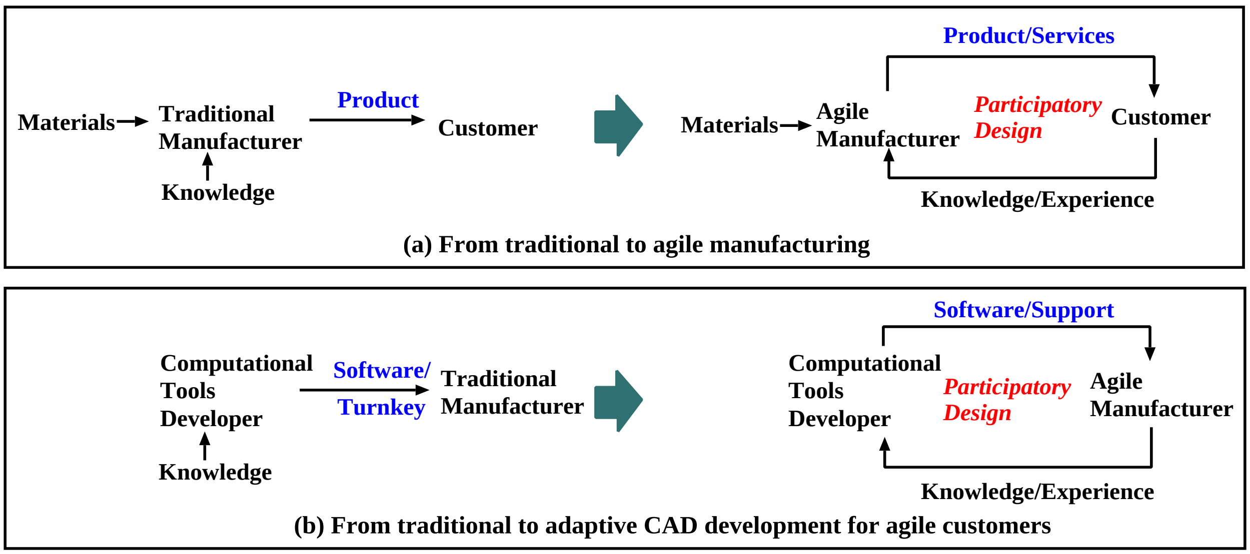

Figure 3: Agile development of software for agile design

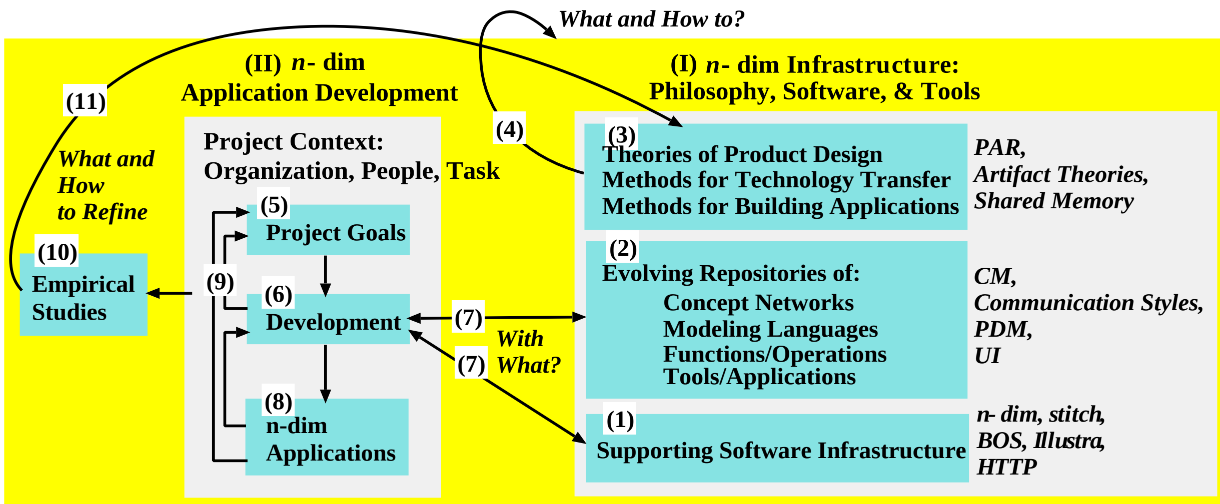

Figure 4: The n-dim project philosophy: iterative study and development

		General Graph	Flat Space	Publication	History	Workspaces (personal and shared)	Tool encapsulation	Scripting Languege	Layered Architecture	NLP tools	Search	Repositories	Design Rationale (e.g., IWEB)
Distributed	1. Time			+	+						+		+
	2. Place			+	+						+		+
	3. Culture	+				+				+			
	4. Languages	+	+			+				+			
	5. Tools						+					+	
	6. Expertise	+					+			+			+
	7. Perspective	+	+			+				+			+
	8. Interaction						+	+				+	+
	9. Usability	+			+		+	+	+			+	
Evolutionary	10. Group learning	+		+	+	+		+	+	+	+	+	+
	11. Relations			+	+	+		+	+			+	+

Figure 5: Influence of n-dim's features on attaining ADIS requirements

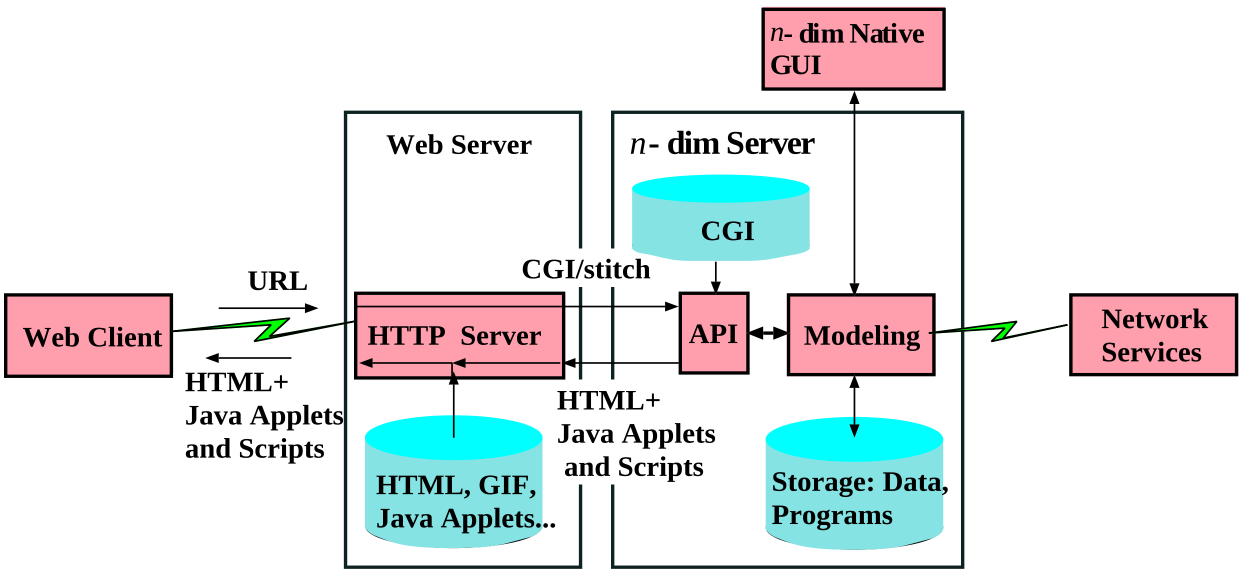

Figure 6: Creating an n-dim server for Internet-based collaboration

Figure 7: The Bug Queue Viewer application

Figure 8: The n-dim model of the Bug Queue Viewer application

Figure 9: The Issue-based discussion application

Figure 10: The n-dim model of the Issue-based discussion application