A Human-Centered Enterprise Information System for Agile Design

Yoram Reich† and the *n*-dim group‡[1]

†Department of Solid Mechanics, Materials and Structures, Faculty of Engineering, Tel Aviv University Ramat Aviv 69978, Israel, email: yoram@eng.tau.ac.il

‡Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA 15213, USA

In *The Proceedings of the 15th SME Israeli Conference on Advanced Technologies in Engineering, Management, and Manufacturing*, July, 1995, p. 264-270.

1 Introduction to agility

We associate *agility* with the ability to (1) quickly detect changing markets; (2) rapidly learn to take advantage of these market changes; (3) detect new techniques, adapt them to the enterprise culture, assimilate them into the enterprise while maintaining their spirit, and use them effectively; (4) meet varying standards in diverse markets (with as little as possible an overhead to the manufacturing process); and (5) be able to customize products to individual preferences. This work deals with the development of an enterprise information system that supports such abilities.

One fundamental property of agility, as seen from the above definition, is the synergy between tools or technologies (may them be advanced or old) and people who can use them effectively within a particular cultural context. Any technology can become merely a pill (Ealey, 1994): its usefulness may fade quickly, and often, its temporary success could be attributed to benign factors, such as extensive temporary organizational awareness to critical issues. The traditional technology "pill"—machines for replacing human manpower—had its successes and failures (when the human factor outweighed the technological impact). The more recent pill—computer tools for replacing human thought or standardizing tools for supporting easy software integration—will fail if not transformed into human-centered support technology.

As a rule, technology will not make the difference in competitiveness through agility. First, technology can be purchased, thus cannot be the underpinning of competitiveness; this holds for enterprise integration tools as a product of technology. It is the *content* of the tools—the enterprise knowledge—that make the difference rather than the tools themselves. Unfortunately, it takes significant effort to construct a usable model of an enterprise knowledge. Second, the development of tools requires intimate understanding of the specific enterprise culture which takes time to acquire and the operation of new technology requires long, rather than quick, learning period. Thus, due to human learning limitations, *there is no agile way to achieve agility.* Rather, it must be planned for in advance.

---

[1]Over the years, many people have contributed to the *n*-dim project. The current *n*-dim group members are: Robert Coyne, Douglas Cunningham, Alen Dutoit, Eric Gardner, Suresh Konda, Sean Levy, Ira Monarch, Robert Patrick, Yoram Reich, Mike Terk, Eswaran Subrahmanian, Mark Thomas, and Art Westerberg.

Japanese practices have shown that easily understood tools that can disseminate everywhere in an organization can outweigh new technology that is not trivial to operate (Ealey, 1994; Goldman et al, 1995). They have preferred common sense to automation by computers as a goal in itself and have distributed responsibility over a task to those who know best about it, i.e., the workers who do the job instead of "experts" that use new computational technology. When we talk about agility we understand that responding quickly may involve forming virtual organizations or teams, each of which with its established areas of expertise, cultural management practices, legacy tools, historical records decision support of previous designs with their, failures, successes, and modifications.

With this background in mind, being aware of the difficulties in introducing new technology, we describe a human-centered enterprise information system for design, called n-dim currently under development. n-dim is designed (1) to be extremely usable and adaptable to workers with different levels of computer-literacy; (2) to support existing practices in a natural manner, including the maintenance and incorporation of existing information and legacy tools; (3) to support the design and incorporation of new tools, practices, and policies; and (4) to support synchronous and asynchronous collaboration of participants.

2 An information integration infrastructure (I^3)

There are three key dimensions of an agile enterprise (Goldman et al, 1995): (1) enriching the customer; (2) reorganizing to master change; and (3) cooperating internally and externally. These dimensions are embedded in a holistic system view that focuses on the role of people, knowledge, and resources, and that assumes that agility is a dynamic, context specific, never ending process.

Enriching the customer means that the customers perceive their relations with the producers as enriching. The nature of products shifts from selling services to solutions (Goldman et al, 1995). Better solutions are created through long-term relationships in which the customers and the producers understand each other. Again, this implies that agility must be planned for and evolved. An ideal form of such cooperation is called participatory design (Reich et al, 1992). *Reorganizing to master change* means the creation of a flexible organization, with flattened managerial hierarchy, where decision-making authority of delegated and distributed. This supports concurrent operations and direct communication between participants. *Cooperating internally and externally* entails changing enterprise and people culture to appreciate the valuable role that it has and the ability to exercise it without damaging self interests.

All these dimensions rely heavily on the human or organizational factors. These are hard to change, inevitable, yet insufficient. Agility requires some *information integration infrastructure* (I^3) to support:

- the communication between customers and producers, and between different producers; in particular, the direct communication between design participants to flatten the hierarchy of information flow;
- to allow flexible enterprise reorganization including the formation of virtual companies and the quick formation of project teams within them; and
- to accumulate and organize the history of a design product and its design to support sustained long-term relationships between customers and producers and

to allow for a quick response to related designs.

An I^3 can be designed such that its users have control over which knowledge they make available for other team members potentially from other companies that make a particular virtual enterprise. Thus, it allows to maintain the interests of the original companies. This is extremely critical since it is the content in the infrastructure that is valuable and not the tool itself.

There are two aspects to building an I^3: the hardware and the software. Most studies dealing with I^3 focus on building a network 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 for graphic or product model, respectively. Many issues are expected to be resolved or avoided due to the introduction of other standards (such as those for trade agreements, forms, libraries, etc.). These studies miss the critical issue that at the design stage, no standards, or even shared terminology exists between participants. It is part of the process to establish them based on mutual agreement. This 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 practices*, is that producers of I^3 will be agile manufacturers and learn to provide solutions (including development and maintenance) for other agile manufacturers, rather than standardized products.

Being an agile manufacturer of I^3 entails adopting a particular view of system development; it is an evolutionary process guided by a methodological emphasis on collaborative prototyping or participatory design (Budde et al, 1992; Reich et al, 1992).

3 *n*-dim: A system for evolving agile design practices

n-dim is a 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 (Levy et al, 1993; Subrahmanian et al, 1991). The development of *n*-dim follows the collaborative prototyping methodology. Supporting this process requires that the *n*-dim environment addresses the problems of asynchronous communication, longitudinal integration of information practices and tools, and multiple perspective reconciliation. Furthermore, *n*-dim must be able to evolve relatively easily when new functionalities are sought.

3.1 How do features of *n*-dim support agile design

There are several main features of *n*-dim that contribute to supporting agile design.

Generalized modeling over a 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; that is, objects do not contain other objects, *per se*. Instead, multiple structures can be imposed on this flat space by means of *models*. Models are comprised of *links*, or relationships between objects (models themselves being objects).

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 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.

Models can also serve as *modeling languages*, that is templates for structuring information. Modeling languages are represented as models themselves (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). Languages for structuring formal knowledge can be used to code expressions, constraints, production rules, and other graph-based models such as conceptual structures (Sowa, 1984). Through the latter models, *n*-dim could easily 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, task allocation, 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, a 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. Design histories can also be charted and organized for continued elaboration and refinement across projects, organizational entities, and cultural contexts.

Publishing is a mechanism for making models formally exchangeable and persistent following the library as a metaphor. 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 control as a means to control knowledge sharing. A published model is inalterable (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 good designs.

Text Processing, Indexing, Structuring, and Retrieval.

Text processing or natural language processing (NLP) enables the discovery of terminological patterns implicit in large text corpora. The discovered patterns provide a basis for organizing, indexing, and retrieving information in various sub-domains – in effect helping to make explicit conceptual structure implicit in the co-occurrence patterns of terminology in large domain-specific corpora. These conceptual structures support the creation of models by indicating the important features a model has to represent as well as the terminology that can be used to label these features. Since the links between these labels and their textual contexts can be maintained, contextualization and documentation of these models can be preserved and easily accessed.

Such context preserving, structuring, and documenting capabilities support: (1) fast introduction of team members to the various perspectives that are being brought to bear in the on-going design of artifacts and (2) the reconciliation of these multiple perspectives by crystallizing their differences and creating a graphic and indexing space in which these differences can be negotiated (Konda et al, 1992).

Integration of CAD/CAM tools and enterprise legacy tools.

The *n*-dim architecture actively aids in the integration of information from CAD/CAM tools. For example, a layout from a particular CAD tool can be related to appropriate information such as design decisions, test data, etc. However, the degree of data and tool integration will be based on the maturity of the tools and the level of integration desired. This can include (1) black-boxes where *n*-dim only allows for the management of inputs into and outputs from the tools in context to (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.

n-dim is developed as a flexible layered architecture. It is developed on top of a prototype-based (a generalization of object-oriented) language that allows to encapsulate code written in diverse languages such as C or Fortran. Thus, enterprise legacy tools can be linked to *n*-dim and start to behave as *n*-dim objects.

Communication facilities including integration of multi-media information.

n-dim has a *talk* facility that provides a means for *synchronous communication*. It allows participants to exchange short notes or arbitrarily complex information such as models.

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 outlined model.

While NLP tools provide facilities for capturing textual context, *n*-dim's modeling facilities also allow for capturing other forms of context such as drawings, photographs, audio, and video. Video (and the associated network) technology is rapidly improving to the state where we should 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.); the development and evaluation of an integrated, on-line recording, teleconferencing, archiving, retrieval facility. Each of these activities is included in our effort to develop and implement collaboration support systems.

3.2 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. Conceptually, a dynamic and flexible modeling environment is required to enable negotiation and consensus promotion among multiple perspectives. n-dim is different from systems that support specific domains – it is designed to be modified and evolved based on the needs of the designers in processes of product creation in their respective domains. This view extends the time honored rapid prototyping tradition of putting the end-user (the designers) in direct collaboration with the developer.

n-dim is also being designed to handle very large applications. To help achieve this, objects themselves are not stored in the database, but only the attributes necessary to search for objects (e.g. the intrinsic attributes). The objects themselves are stored locally in individual workspaces, or (possibly) in shared spaces if they have been published (and thus made immutable). 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 utilized to great advantage in scaling n-dim up 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). Studies will be conducted for the appropriate configuration based on network and load balancing requirements.

3.3 Agile design with customers: participatory design

By and large, most 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. No such assumption about the proficiency of design participants (customers as well as designers) with computers can be made with respect to participation in agile design. Therefore, in order to build an I^3 we need to attend to the *usability* issue.

It is critical that computer tools be usable by design participants. Furthermore, as observed in many studies about the introduction of software into organizations, such support tools may not be congruent with the way designers or customers think. In such cases, tools remain unused. Our approach to the development of computer tools—participatory design and evolutionary prototyping—is geared towards alleviating this problem (Reich et al, 1992). Participatory design is predicated on the use of the expertise of designers and customers, on the development of long-term relationships in which both the designers and the customers understand each others needs and cooperate to create an evolving product that will benefit both. Thus, participatory design can serve as a foundation for agile design.

4 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.

Subsequently, we described the n-dim architecture developed for collaborative design, but whose philosophy of how design work is done, and how it can be evolved continuously — collaborative prototyping and participatory design — make it an ideal I^3 for agile design. In the exposition of n-dim we stressed the facilities for diverse information modeling, communication, text and models analysis, integration of legacy tools, and the ability to scale n-dim to real applications.

References

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

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

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

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

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.

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.

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., 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.

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.