Proceedings of the CHI 2004 Workshop on Designing for Reflective Practitioners

David Redmiles
University of California, Irvine
redmiles@ics.uci.edu

Kumiyo Nakakoji
University of Tokyo
kumiyo@kid.rcast.u-tokyo.ac.jp

Anders Mørch
InterMedia
University of Oslo
anders.morch@intermedia.uio.no

Gerhard Fischer
University of Colorado at Boulder
gerhard@cs.colorado.edu

April 2004

ISR Technical Report # UCI-ISR-04-2

Proceedings of the CHI 2004 Workshop on Designing for Reflective Practitioners

Edited by

David Redmiles

Institute for Software Research and Department of Informatics
University of California, Irvine
Irvine, CA 92697-3425 USA
redmiles@ics.uci.edu

Anders Mørch

InterMedia
University of Oslo
NO-0318 Oslo
Norway
anders.morch@intermedia.uio.no

Kumiyo Nakakoji

University of Tokyo
4-6-1 Komaba,
Meguro, Tokyo,
153-8904 Japan
kumiyo@kid.rcast.u-tokyo.ac.jp

Gerhard Fischer

Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80303
USA
gerhard@cs.colorado.edu

ISR Technical Report # UCI-ISR-04-2

April 2004

Abstract: Donald Schön described professionals as practicing reflection-in-action. This characterization inspired many researchers to experiment with computing systems whose interfaces supported and even prompted reflection on the part of end users. Many parallels to Schön's notion exist in different communities whose members attend CHI. Some work extends beyond computer interfaces to social and organizational issues. This workshop was an opportunity for diverse researchers to come together to identify and trace the evolution of common threads, to share and assess solutions, and to open channels of communication that will support one another's long-term objectives of designing for reflective practitioners.

Table of Contents

Organizers Welcome and Introduction

Designing for Reflective Practitioners: Sharing and Assessing Progress by Diverse Communities

Gerhard Fischer
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80303
USA
gerhard@cs.colorado.edu

Anders Mørch
InterMedia
University of Oslo
NO-0318 Oslo
Norway
anders.morch@intermedia.uio.no

Kumiyo Nakakoji
RCAST
University of Tokyo and PRESTO, JST
4-6-1 Komaba, Meguro, Tokyo, 153-8904 Japan
kumiyo@kid.rcast.u-tokyo.ac.jp

David Redmiles
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3425
USA
redmiles@ics.uci.edu

WORSHOP THEME AND GOALS: DESIGNING FOR REFLECTIVE PRACTITIONERS

Donald Schön described professionals as practicing reflection-in-action [13]. This characterization inspired many researchers to experiment with computing systems whose interfaces supported and even prompted reflection on the part of end users [5]. Many parallels to Schön's notion exist in different communities whose members attend CHI. Some work extends beyond computer interfaces to social and organizational issues. This workshop is an opportunity for diverse researchers to come together to identify and trace the evolution of common threads, to share and assess solutions, and to open channels of communication that will support one another's long-term objectives of designing for reflective practitioners.

Author Keywords
Reflection-in-action; software critics; software agents; situated action; participatory design; open source.

ACM Classification Keywords
D.2.2 Design Tools and Techniques; H.5.2 User Interfaces; K.4 COMPUTERS AND SOCIETY

PARALLEL THEMES WITH A COMMON CHALLENGE

The theme and title for this workshop is inspired by Donald Schön's writings about the reflective practitioner in which he describes professional practice as transcending technical rationality [13]. Ill-formed problems lead to breakdowns, which become opportunities for reflection and modification of practice. Many others have articulated related concepts, and concerns. For instance, Fred Brooks distinguished between accidental and essential activities for designers of software systems [1]. Software tools could support mundane aspects of designers' work, but the most creative aspects would still elude computer support. Herbert Simon also referred to the bounds of rationality and evoked the anecdote of the painter faced with a blank canvas to describe ill-formed problems that required a different kind of thinking [14]. Designers postulate starting points, evolve them to stable substrates, and then rethink them. Lucy Suchman demonstrated the limits of rationalized designs in her seminal characterization of situated action [15]. As she notes, anticipating all potential user behaviors is not a feasible approach to design.

Interestingly, these concepts create a conflict of sorts for researchers in computing. Namely, if computer software operates on the plane of technical rationality, how can it support reflective and situated action by practitioners in the real world?

TAKING UP THE CHALLENGE

The challenge of designing computer support for reflective practitioners has been taken up by many communities and from many perspectives. There are software-based approaches, cognitive approaches, and social and organizational approaches.

Among the software based responses, software critics are intended directly to trigger reflection by end users, providing feedback on design tasks while designers are still in the context of making design decisions [5]. Critics are not intended to replace human decision making, but to complement it [7][16]. Similarly, software agents proactively coordinate to assist end users, including software designers as end users [9]. Even techniques for supporting software process descriptions have evolved from rigid prescriptive systems to reflective models that can adapt to exceptions [11].

There are cognitively based responses to supporting reflection. In a sense, the notion of affordances [10] and even social translucence [4] may be interpreted as styles that enable essential reflection by removing the distraction of an awkward interface.

CHI 2004, April 24–29, 2004, Vienna, Austria.
ACM 1-58113-703-6/04/0004.

There are also social responses to this challenge. The computer-supported collaborative learning community seeks to enhance reflection by integrating working and learning, physical and computational artifacts, and different communities of interest [6]. The methods and techniques of participatory design integrate end users into the design process to achieve greater realism in systems [2][8]. The open source movement might also be interpreted as a style of software development geared toward placing the evolution of a software system with the practitioners [12]. Activity theorists also emphasize the role of reflection in community activity [3].

FOSTERING A NEW COMMUNITY

The above responses by different communities to the reflective and situated nature of work practice have existed for decades, and have faced trials and refinements. Many themes have evolved within these disciplines. Some have appeared independently under different terms and in different settings. The purpose of this workshop is to bring together representatives of diverse communities who have designed solutions that support reflection-in-action, or related notions such as those named above. The organizers seek to trace the evolution of common threads, to share and assess solutions, and to open channels of communication that will support one another in the long term. We seek to foster a sense of community among diverse researchers who all have been designing for reflective practitioners.

RELATION TO CHI 2004 THEME

The Conference Overview by Elizabeth Dykstra-Erickson and Manfred Tscheligi describes the themes of CHI 2004 as forming connections and expanding boundaries (see http://www.chi2004.org/geninfo/overview.html). In this workshop, we seek to build connections among people from many disciplines and strengthen communication in the long term. The theme of focusing on reflective practitioners has a long history because it is a problem rooted in humanity, namely the abilities and instincts human beings have for carrying out activities in a complex world, where routine action is frequently frustrated. The major change in recent decades is the involvement of the computer in these activities. From one perspective, the computer is merely a new opportunity for understanding humanity. In this sense, the workshop is also forward-looking as well.

REFERENCES

1. Brooks, F. No Silver Bullet: Essence and Accident in Software Engineering. *IEEE Computer,* 20(4), 10-19, 1987.
2. Ehn, P. Work-Oriented Design of Computer Artefacts, Arbetslivscentrum, Stockholm, Sweden, 1988.
3. Engeström, Y. Coordination, Cooperation and Communication in the Courts, in *Mind, Culture, and Activity,* Cambridge University Press, 369-388, 1997.
4. Erickson, T., Halverson, C., Kellogg, W., Laff, M., Wolf, T. Social Translucence: Designing Social Infrastructures That Make Collective Activity Visible. *Communications of the ACM,* 45(4): 40-44, 2002.
5. Fischer, G. Domain-Oriented Design Environments, *Automated Software Engineering,* Kluwer Academic Publishers, Boston, MA, 177-203, 1994.
6. Fischer, G. Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems, *Proceedings of the 24th IRIS Conference* (eds: S. Bjornestad, R. Moe, A. Morch, A. Opdahl), Ulvik, Department of Information Science, Bergen, Norway, 1-14, August 2001.
7. Fischer, G., Nakakoji, K. Beyond the Macho Approach of AI: Empower Human Designers - Do Not Replace Them, *Knowledge-Based Systems Journal, Special Issue of AI in Design,* 5(1), 15-30, 1992.
8. Greenbaum, J., Kyng, M. (eds.), Design at Work: Cooperative Design of Computer Systems, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
9. Hilbert, D., Redmiles, D. Large-Scale Collection of Usage Data to Inform Design, *Eighth IFIP TC 13 Conference on Human-Computer Interaction* (INTERACT 2001, Tokyo, Japan), 569-576, July 2001.
10. Norman, D. The Design of Everyday Things, Currency-Doubleday, New York, NY, 1989.
11. Nutt, G. The Evolution Towards Flexible Workflow Systems, *Distributed Systems Engineering,* 3(4), 276-294, December 1996.
12. Raymond, E. S. The Cathedral and the Bazaar, *First Monday,* 3(3), 1998.
13. Schön, D. The Reflective Practitioner: How Professionals Think in Action, Basic Books, New York, 1983.
14. Simon, H. The Sciences of the Artificial, The MIT Press, Cambridge, MA, 1981.
15. Suchman, L. Plans and Situated Actions, Cambridge University Press, Cambridge, UK, 1987.
16. Terveen, L. An Overview of Human-Computer Collaboration, *Knowledge-Based Systems Journal, Special Issue on Human-Computer Collaboration,* 8(2-3): 67-81, 1995.

Session: Communication and Sense-Making

A Meta-Communication Model for Reflective Practitioners

Fahri Yetim
Information Systems Department, College of Computing Sciences,
New Jersey Institute of Technology, Newark, NJ 07102- 1982, USA
Email: *Fahri.Yetim@njit.edu*
http://web.njit.edu/~yetim

Position Paper: CHI 2004 Workshop: "reflective practitioner"

Developing any kind of information system embodies reflections about the desired features of the resultant system. The reflective practice becomes more important the more the differences in technologic standards, social values, norms, assumptions and interests, etc. in global contexts interfere the sphere of the Information Systems Development (ISD). To deal with such issues and underlying validity claims in a rational and reflective way, previous approaches to rational and reflective practice in ISD have already emphasized that a rational practice requires not only knowledge and its successful transformation into efficient and effective action but also justification of normative implication for those involved and affected.

I have extended the framework for reflective practice proposed by Ulrich (2001) by integrating – among others – discourse-ethical concepts advanced by J. Habermas (Habermas 1984, 1996) and suggested a model of meta-communication for reflective practice, which provides a wider spectrum of concepts for dealing with global challenges in a rational and reflective way. The operationalization of the model towards the practice is illustrated by the concept of communicative genres (referred as to 'communication action patterns). The argument is that meta-communication processes guided by discourse-ethical principles promote a legitimate definition, design, and development of such patterns, and thus increase the legitimacy of resultant norms and contents of patterns for communication, especially in intercultural interaction contexts (Yetim 1998).

In my approach (Yetim 2004), I distinguished between three different types of meta-communication:

- Ex ante meta-communication (taking place before action),
- Meta-communication in action (taking place during action), and
- Ex post meta-communication (taking place after action).

The meta-communication model itself consists of two levels:

- Clarification level (where conversation for clarification takes place). At this level there are eleven clarification issues to be reflected on.
- Discourse level (where the discursive examination of contested claims takes place). At this level, there are eight discourses, which are related to the clarification issues.

This diversification also allows us to easily relate the discourse ethical differentiation of discourses of justification to ex ante meta-communication, and discourses of application to meta-communication in action. Ex post meta-communication remains related only to breakdowns that occur when an action has taken place.

The approach contributes to the advancement of the previous research dealing with reflective practice by providing additional concepts. These concepts allow renegotiations of system features and thus can support the continuous co-evolution of a system.

References

Habermas, J. (1996). *Between Facts and Norms*. Cambridge, MA, MIT Press.

Habermas, J. (1984). *The Theory of Communicative Action: Reason and the Rationalization of Society*. Boston, MA, Beacon Press. (Vol. I)

Ulrich, W. (2001a). A Philosophical Staircase for Information Systems Definition, Design and Development. *Journal of Information Technology Theory and Application* 3 (2001), 55-84.

Yetim, F (2004). Meta-Communication for Reflective Practice in Information Systems Development: A Discourse-Ethical Approach and its Operationalization by Patterns of Global Communication. To appear in: *Information and Organization*.

Yetim, F. (2001): A Meta-Communication Model for Structuring Intercultural Communication Action Patterns. *SIGGROUP Bulletin* 22(2), 16-20. (Reprinted from the Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modeling.

Yetim, F. (1998). Interkulturalität und Informatische Gestaltung – Eine Interdisziplinäre Annäherung. *Informatik-Spektrum* 21 (1998), 203-212.

Making the common object of a work "visible" and reflectable: a case of emerging software product

Kari Kuutti & Tonja Molin-Juustila
University of Oulu, Department of Information Processing Science
(kari.kuutti@oulu.fi, tonja.molin-juustila@oulu.fi)

Introduction: a need for a "visible" object of work

The most fundamental point of reflection for a reflective practitioner is the purpose of the work involved: are we doing the right thing? The cultural-historical activity theory (CHAT, see e.g. Kuutti 1996), one of the frameworks which have gained interest wihin the HCI research during the last years, has a good concept to deal with this, the concept of object of a work. activity. One of the foundational hypothesis of the CHAT framework is the idea, that when seen from the viewpoint of an individual actor, work and also other spheres of life is organized into activities which are largest meaning-giving units within an individual horizon. According to the theory, activities are separated from each other by their object, which is the purpose of the corresponding activity: a potential fulfillment of a need that can be reached from where we are now, when we organize ourselves accordingly, select the right tools and transform the world by a set of directed, interconnected actions leading towards that fulfillment.

If the object of a work activity is well known by participants, it is a good beacon: at every moment everybody can check and compare the current state of affairs with the object and reflect and correct the direction, if needed, and thus there is a strong self-organizing feature in the arrangements. Correspondingly, if the object is known only by some, but it is not shared and common, there is a considerable effort in communicating the object, and the work has to be more strictly organized from above to ensure that the direction is right and the intermediate results will serve their ultimate purpose (the popularity of the terms of "mission" and "vision" in current management literature are clearly related to this issue).

It is, however, characteristic to human life, that objects cannot be easily "given " from above or from outside, but they must be "invented" or appropiated by the actors themselves, otherwise they will not have such organizing power – the given outside object, and the corresponding activity will exsist only on paper, and the participants will be engaged into something else instead. Because objects and activities cannot be given, they are often not very clear to participants, and even when they may be clear for a while, they are not fixed. When situations develop and unfold in the course of actions, new possibilities and limitations reveal themselves, and the object will evolve accordingly. In times of crisis, an evolution is not enough, but it may be necessary to change the object radically. No wonder, that a considerable slice of work in organizations is, in a way or another, devoted to figure out the current object of work and communicate it to other stakeholders. Working out the object of work is a major point of reflection within work organizations.

It would be helpful, if we had ways of making the object more "visible", comprehensible, and even manipulable, and thus also a better tool for reflection. And given the prominence of information technology at the workplace one would like to see, if it could help us in this. We will illustrate the issue with a practical example, where different communities of stakeholders needed to produce a common conception of their object. The case is related to an ongoing doctoral thesis work in Oulu.

Case: New product development in an IT company

The ideas presented here are based on several years' fieldwork, observation and experiments in a medium-sized software company. The case company operates in software product business. It has sales offices, development centers and partner companies employing about 1000 people worldwide. The business consists of few different products for well-defined markets. Our analysis has been concentrating on their one new product idea and the emerging business for that (during our cooperation the unit consisting of about 100 people). The new software was more like an enterprise solution type of product than a true mass-market package software.

The development of the new product run into problems. The market and needs of the new innovation were not so clear in the case company and the software product itself was much more iteratively produced than it was assumed. The product seemed to be in a constant design phase; the very early releases were implemented for the pilot customers and later on their applications were updated based on the new releases. Both the company and its sales partners based their businesses on these early customers. Company was trying to understand the customers and generalise the market needs from these early pilots. At the same time sales partners were demanding new properties to satisfy their prospects. There was an obvious danger that the company long-term visions and the pilot customers needs were not always in harmony with each other, but there seemed to be no way to reconcile them.

With the iterative nature of development work, there was a potential to reflect the long-term visions with the market. Ongoing interplay between the representatives at the customer interface and the product developers would be the prerequisite for this. However, the product alone was not enough as a common focus for their interaction. Also the dispersed information about the business assumptions of the different stakeholders needed to be shared. However, this was lacking a suitable means for communication. There was a need to keep the different functional actions better focused during the early iterations in order to find the best possible business for the new product. We found a special piece of knowledge that needed to be shared among the different stakeholders around the new business area – a shared vision about the product-market-user -combination. As stated in one of the workshops, there was a need for "more systematic way of combining the strategies, technology, vision and the requirements as to company targets" (memo 12.10.2000). In short, what was missing was a new cross-functional work activity of defining the new product, the emergence of which was made difficult by different visions each function had about the potential object of the activity and also their fluidity and constant change.

Without a common frame of reference it was impossible to understand "how to evaluate and efficiently utilise the current implementations against these targets". In order to define the best possible fit between the product and the user/market needs, they not only needed to share their assumptions about this, but they also needed to reflect these assumptions against their everyday practice. The "shared vision about the product-market-user –combination", is a hypothetical and tentative idea that evolves all the time when more experience is gained.

As a result we did build our tentative model for the interaction between the different stakeholders. In our market centred approach to product innovation development (MAPID) the new software product development is seen as a process of iteratively both to identify the market needs and to improve a product to match them, cf. Figure 1. Next is an illustration of the main elements of the model.

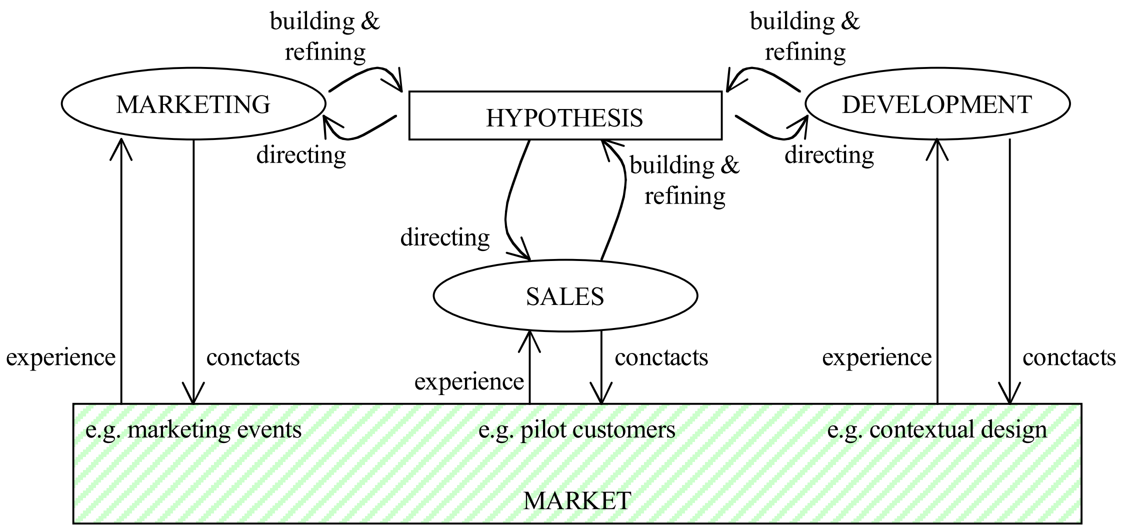


Figure 1. The MAPID model.

Creation of the product-market-user vision is seen as a hypothesis of the existence of a market and a potential product for this market – an emerging object for a new product business activity. The hypothesis (or there might be several of them) will be a combined view of the different stakeholders: the shared assumptions and the generalised view from the early contacts with the field (e.g. pilot customers). Developers need to understand how the marketing and sales people see the future business and what are their needs for the development. Marketing and sales people need to understand what kind of solutions the development is able to offer for them now and in the future.

The hypothesis evolves over time driven by the actions at the customer/market interface. Daily contacts with the possible customers (marketing and sales events, customer pilot projects, contextual design etc.) are the main points for the evaluation of the current hypothesis. The hypothesis is both directing the everyday actions of different stakeholders, and constantly questioned, validated and refined based on the experiences from the field. When developing a new software product by the means of pilot customer projects, each customer case should be treated as an experiment to validate and correct the current hypothesis. The concrete information about the real customer case should be contrasted with the assumptions described in the hypothesis.

We implemented a practical version of this model in the case company during 2002-03. Initially we had fancy ideas of rich IT support for both the object/hypothesis itself and the interaction around it, but we came soon into our senses and instead decided to produce something that could be taken into immediate use; without any further training, using already available tools and means for communication, and with minimal additional workload to our stakeholder users. We ended up with a true technology mundane solution of pre-structured Word-templates, predefined shared directories, e-mail facilities, and some organizational arrangements – a new committee, and some guided practices for producing and reflecting these templates, cf. Figure 2.

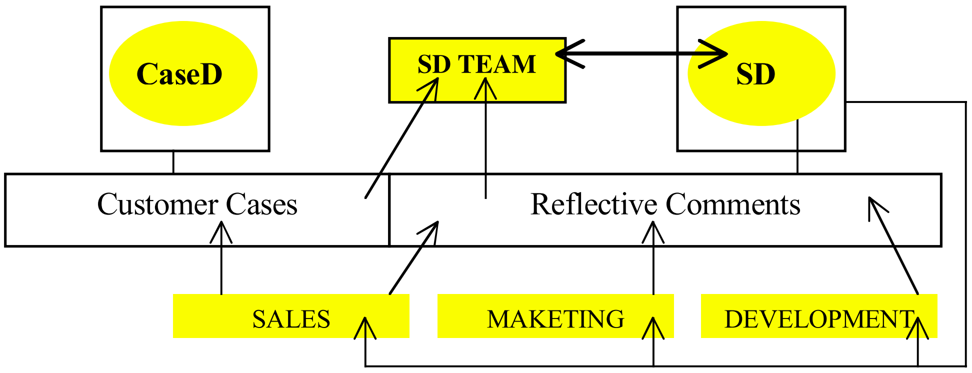


Figure 2. The implementation of the MAPID model in the case company.

The main document template (called Segment Description; SD in short) is used in documenting and sharing the hypothesis. It needed to be concrete enough to help to focus their daily efforts. Each SD has an owner appointed by the company management. The owner is responsible to collect a cross-functional team to produce the SD. The team would use all the available information to build a new SD. All stakeholders are able to comment the current SD based on their daily contacts with the field, and that way support the evolution of the SD. Comments need to be saved for further refinement of the SD. There is also a template for documenting the early and real customer cases (called Case Description; CaseD in short). This template was following the structure of SD in order to ease its reflection. The owner of the SD is responsible for the management of the teams work and the iterative progress of the SD.

In line with the CHAT assumptions, we did not invent the model by ourselves and give it to the organization "from above", but it was co-invented and constructed during a longer period. Thus the results were also accepted very well by the organization, and it started to become part of their thinking and vocabulary. When the system was set to work, it seemed to start to produce such interactions we have been hoping for. However, our project funding ended in early 2003, and we have not had possibilities to follow clearly, if our work has had any lasting effects. After summer 2003 we have not heard anything from the company (mainly because the main researcher, second author, has been busily writing her thesis). We hope that we can update the situation for the workshop.

Kuutti, K., *Activity Theory as a potential framework for human-computer interaction research*, in *Context and Consciousness: Activity Theory and Human Computer Interaction*, B. Nardi, Editor. 1996, MIT Press: Cambridge. p. 17-44.

Virtual Workshops to Support Reflection in Action

Maria Francesca Costabile[1], *Daniela Fogli*[2], *Piero Mussio*[3], *Antonio Piccinno*[1]
[1] Dipartimento di Informatica, Università di Bari, Bari, Italy
{costabile, piccinno}@di.uniba.it
[2]Dipartimento di Elettronica per l'Automazione, Università di Brescia, Brescia, Italy
fogli@ing.unibs.it
[3] Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano, Milano, Italy
mussio@dico.unimi.it

Position paper

In our experience of computer scientists, we cooperate in participatory projects to develop computer systems to be used by professional people, such us medical doctors, geologists, mechanical engineers. These professionals need to use computer systems for performing their work tasks exploiting all the communication and operation possibilities offered by these systems, but they are not and do not want to become computer experts. This has motivated the definition of a particular class of end-users, that we call *domain-expert users* (or *d-experts* for short) [4]: they are experts in a specific discipline (e.g. medicine, geology, etc.), not necessarily experts in computer science, who use computer environments to perform their daily tasks. These d-experts often complain about the systems they use, they feel frustrated because of the difficulties they encounters interacting with them.

In domains of their competence, communities of d-experts progressively developed documentation styles, notations and procedures to record the community's knowledge - abstract or concrete concepts, prescriptions, results of activities - as documents. This enabled the community's knowledge to be available to members when and where they require it and in the form required to perform their current activities. Notations developed by the communities of experts determine the layout and appearance of the document thus permitting the expression of tacit information – embedded and conveyed by the document shape as well as by images, icons, textual words, which are meaningful only for experts in the domain at hand. Documents expressed in these notations support reasoning based on implicit knowledge, namely the knowledge that people possess and currently use to carry out tasks and to solve problems but that they are unable to express in verbal terms and that they may even be unaware of. It is a common experience that in many application fields experts possess a large amount of implicit knowledge, since they are often more able to do than to explain what they do. Implicit knowledge depends on the specific work domain and is related to the d-experts "practical competence and professional artistry in achieving a task" [8]; it is exploited by users to interpret the documents and, nowadays, to interpret messages from the computer systems.

As designers, we are challenged to develop interactive software systems that a) support their users in exploiting their practical competence, and b) enables the practitioner to develop and extend the knowledge available to the profession [1]. To develop such systems, we recognize the importance of notations developed by d-expert communities as reasoning, communication, and documentation tools, and we adopt a methodology for developing virtual environments, in which users interact using languages that are a formal representation of their traditional notations and virtual tools that recall the real ones with which users are familiar. More specifically, the methodology takes into account the following observations:

1) We recognize user notations and 'semiotic systems' as tools to convey user tacit information. The notations developed by the user communities from their working practice are not defined according to computer science formalisms but they are concrete and situated in the specific context, in that they are based on icons, symbols and words that resemble and schematize the tools and the entities used in the working environment. Such notations emerge from users' practical experiences in their specific domain of activity. They highlight those kinds of information users consider important for achieving their tasks, even at the expense of obscuring other kinds, and facilitate the problem solving strategies, adopted in the specific user community [2].

2) We recognize that new computer-based reasoning and communication modalities created the possibility of new modalities of communication and of the development of completely new 'semiotic systems' and notations. We stress that d-experts, using systems that exploit these new semiotic systems and notations, must master them in order to maintain the interpretative expertise of the virtual world in which they operate. Sometimes, the new modalities diverged so radically from the past that large portions of the users' practical experience failed to generalize to the new situation [5]. The current phase of introducing digital media coupled with computerization poses yet a more fundamental

challenge to user work as a whole. The change in the material and technological mediation from traditional to electronic media suggests a drastic and through-going reorganization of everyday work practice.

3) We recognize the need of local categorization of knowledge. Our view refines the Schön observation that "the categorization of knowledge in terms of a category like 'tool', as distinct from the ordinary, familiar coherences of objects as they go together in our everyday life, is what I mean by the formal categorical character of knowledge. And it is one of the key features that separates schools from life. The ways in which things are grouped together, the way in which things are treated as similar and different, are not the way in which they are grouped and treated as similar and different in our ordinary life experiences" [9]. We stress that in ordinary life experience, experts use different categorizations of events and things according to the current activity they are developing. We observe that these categorizations are reflected in the experts' notations and semiotic systems; moreover, different categorizations of events and things linked to the specific culture of the expert and to the current context occur. These different categorizations lead to the existence of different notations and semiotic systems – mechanical engineers document their activities in a different way from physicians - and to the existence of dialects within notations and semiotic systems - mechanical engineers in Italy use different notations than their colleagues in other countries.

Starting from these observations, three principles are at the basis of our methodology to design interactive software systems: i) the language in which the interaction with systems is expressed must be based on notations and dialects traditionally adopted in the domain; ii) systems must present all and only the tools necessary to perform the user work, without overwhelming users by unnecessary tools and information; iii) systems must present a layout simulating the traditional layout of the tools employed in the domain, such as for example mechanical machines or paper-based tools.
Our approach to the design of a software system devoted to a specific community of domain-expert users is to organize the system as composed of various environments, each one for a specific sub-community. Such environments represent *virtual workshops* [3][4] since they are organized in analogy with the artisans workshops, where the artisans find all and only the tools necessary to carry out their activities. In a similar way, a d-expert using a virtual workshop finds available all and only all the tools required to develop his/her activities. These tools must be shaped and must behave so that to be usable by the d-expert in the current situation.

In each virtual workshop, d-experts of a sub-community interact using a computerized version of their traditional languages and tools; they get the feeling of simply manipulating the objects of interest in a way similar to what they might do in the real world. In other words, our approach provides each sub-community with a personalized workshop. In this way, d-experts of a sub-community work out data from a common knowledge base and produce new knowledge, which can be added to the common knowledge base, increasing the community knowledge.

Thus d-experts may work cooperatively to reach a common goal; in this sense, the computer system becomes a collaboratory, as defined in [10]: "a center without walls, in which researchers [in our case professionals] can perform their research [work] without regard to geographical location, interacting with colleagues, accessing instrumentation, sharing data and computational resources, and accessing information in digital libraries".

An important activity on which d-experts' collaboration is based is the annotation of documents [6][7]. In the workshop methodology, electronic annotation is a basic operator, on which the communication among different d-experts and the production of new knowledge are based. An expert has the possibility of performing annotations of a piece of text, of a portion of an image or of the same workshop in use in order to extend, make explicit his/her current insights - on the problem at hand or even on the features of the workshop. Annotations are added to the common knowledge base and become accessible by other d-experts, each one accessing the data through his/her own workshop and interacting in his/her own professional language. Such annotations provide further possibilities to support the d-expert to reflect on his/her activities, and to make his reflection available to the whole community. Indeed, the activity of a d-expert is influenced by the observations performed and annotated by a colleague, which are then visible to him/her.

To make an example of how these systems can allow cooperative work of professionals who perform a common task, and how the annotation is important for triggering user's reflections, let us briefly consider a scenario taken from the medical domain. The scenario refers to some physicians collaborating to achieve a diagnosis [4]. A pneumologist and a radiologist incrementally gain insight into a case by successive interpretations and annotations of chest radiographies, performed in (possibly) different places and at (possibly) different times. They work in two different workshops that share a knowledge repository. They achieve the diagnosis by updating the knowledge repository after each session of interpretation of the results and of annotation of their new findings. Working in his workshop, the radiologist is analyzing a chest radiography and recognizes an area of interest denoting a pleural effusion; he then selects from a toolbox the tool that allows him to draw a close curve around the area of interest, and adds to this area a textual annotation that describes its observations about a 'Pleural effusion' that he wants also to communicate to the pneumologist. The system is able to associate a widget to this annotation. This newly created widget will also appear to the pneumologist when he looks at the same radiography in his workshop. By clicking on this widget, the pneumologist may read the radiologist' annotation, that becomes a trigger for his reflective activity.

References

[1] Alexander, I., Book Review: The Reflective Practitioner – How professionals think in action, online <http://i.f.alexander.users.btopenworld.com/reviews/schon.htm.

[2] Carrara, P., Fogli, D., Fresta, G., Mussio, P. Toward overcoming culture, skill and situation hurdles in human-computer interaction. *Int. Journal Universal Access in the Information Society*, 1(4), 288-304, 2002.

[3] Costabile, M.F., Fogli, D., Fresta, G., Mussio, P., Piccinno, A. Computer Environments for Improving End-User Accessibility. *Proc. of 7th ERCIM Workshop "User Interfaces For All"*, Paris, 187-198. 2002.

[4] Costabile, M.F., Fogli, G., Mussio, P., Piccinno, A. End-User Development: the Software Shaping Workshop Approach. Submitted to book *End User Development,* Lieberman H., Paternò F., Wulf V. (Eds), Kluwer Academic Press.

[5] Karasti, H., *Increasing Sensitivity Towards Everyday Work Practice in System Design*, PhD thesis, University of Oulu, Oulu, 2001.

[6] Mackay, W. E., Augmented Reality: Linking real and virtual worlds – A new paradigm for interacting with computers, *Invited paper, Proc. of AVI '98*, L'Aquila, Italy, May 1998, 13-21.

[7] Mussio, P., E-Documents as tools for the humanized management of community knowledge, *Keynote Address*, to appear in *ISD 2003 Proc*., Melbourne, August 2003.

[8] Schön, D., *The Reflective Practinioner – How Professionals Think in Action*, Basic Books, 1983.

[9] Schön, D., Educating the Reflective Practitioner, Meeting of the American Education Research Association, Washington, DC, 1987.

[10] Wulf, W. A., The Collaboratory Opportunity, *Science: New Series*, Vol. 261, 1993.

Sensemaking and Design Practices in Large-scale Group-to-Group Distance Collaboration

Gloria Mark
Department of Informatics
University of California, Irvine
gmark@ics.uci.edu

Steve Abrams
Department of Informatics
University of California, Irvine
sabrams@ics.uci.edu

A new paradigm in collaborative interaction is arising. Large-scale collaborations across distance are becoming more common enabled by technological development such as the Access Grid and the need to bring together not just individuals, but entire groups of experts to solve complex problems. Despite this growing trend, this form of collaboration have not received much attention. In this paper we describe how this new kind of interaction order affects collaboration in the domain of space mission design.

Intersubjectivity, sensemaking, and group interaction

In group-to-group distance collaboration, entire groups, each working in a common space, are connected together through some combination of technologies. People are interacting in multiple social worlds simultaneously: their collocated team, and the larger, distributed team. Interaction in these different social worlds is characterized by different types of sensemaking, where people interpret cues, negotiate, apply expectations, and commit to decisions [4]. In any collaborative context, through the experience of interacting with another, and making sense of the environment, a sense of common meaning, or *intersubjectivity* is developed. Intersubjectivity refers to a state of interaction where perspectives can be mutually or reciprocally understood [3]. Especially sharing a common environment or "community of space", where people directly experience each other, creates favorable conditions where intersubjectivity can emerge. People are constantly modifying their understandings of the other, and consequently are continually constructing shared meanings. In the case of social relationships that are not face-to-face, one understands the other through an "ideal type". Schutz [3] describes that people rely on assumptions to construct a "shared interpretive scheme" (pg. 229). In distant interaction, one receives little or no feedback as to whether one's assumptions about the partner type were accurate. Compared to the full spectrum of possible experiences that can be shared in face-to-face settings, this is meager information. Without information to contradict or update it, distant partners generally continue to rely upon the ideal type.

Interaction does not always remain at the same "level" according to Wiley [5] who frames interaction from the individual to the societal and cultural level. Through interaction, individual meanings can merge into intersubjective meanings, which in turn can emerge into a *generic subjectivity*, which constitutes social structure[1]. Intersubjective interaction over time evolves into "interlocking routines and habituated action patterns" (Wiley, pg. 74) between individuals that can be taken for granted and which affords a degree of predictability to the interacting individuals.

When actors are distributed across distance with technology-mediated interaction, intersubjectivity can emerge differently than in a face-to-face environment. In a collocated setting, it is easier to understand when intersubjectivity is slipping away due to the rich availability of feedback. When generic subjectivity emerges, this is also easier to maintain in a collocated setting as the extent to which people follow (or don't follow) scripts is highly visible. In a distributed setting, with limited feedback through distinct channels (i.e. audio, video, images, text) the "ideal type" perception must be overcome for intersubjectivity to emerge. Experiencing distant behavior through limited social bandwidth makes it difficult to predict routines and patterns, which also can inhibit the development of generic subjectivity. Interaction may also vascillate between levels, e.g. between intersubjective and generic subjective states. No interaction is purely one form of (inter)subjectivity or the other.

The study setting

We performed an ethnographic investigation, guided by our research question of examining different types of sensemaking in group-to-group collaborative settings. We studied a large distributed technology organization, who

[1] Wiley describes four levels, the individual, the intersubjective, the generic subjective, and cultural, but only the middle two levels are treated here as they are relevant for the current study.

researches, designs, and develops space-based scientific technologies and missions. We observed a design team from this organization, comprised of four engineering groups (teams 1–4) distributed around the U.S. Team 1 had 24 team members at Site 1 on the west coast, team 2 had 12 members at Site 2 in the Midwest, team 3 had 9 members at Site 3 in the south, and there was a single person at Site 4 in the southwest. Most of the people on teams 1, 2, and 3 had previously worked together within their teams but had never worked with the other teams in the past. The purpose of collaborating together was to combine different specializations to work on a conceptual design for an actual space mission.

The design team relied on a number of technologies to share design data, audio, and video streams. *NetMeeting* shared applications across sites projecting document views from Microsoft *Excel* and *PowerPoint*. *ICEMaker* [2] linked workstations and shared data, thus enabling the members of the design team to publish design specifications and parameters relevant to a particular subsystem as either numeric data in the spreadsheets. A dedicated person managed the updating of spreadsheets and the projection of spreadsheets both locally and remotely. A *video-teleconferencing* (VTC) service shared the audio of all four sites, and switched the video such that it displayed the view of the recent most vocally active site to the other sites. Multiple large public displays (12 x 6 feet at Site 1 and 6 x 5 feet at Sites 2 and 3) showed the video and the shared applications. *MeetingPlace* managed distributed small group discussions, or sidebars, by sharing multiple voice streams by telephone. E-mail and fax, were also available.

The design team collaborated for a total of nine hours, spanning three days within a week. Three researchers traveled to Sites 1–3 and observed the teams' interactions for the whole duration. We videotaped the teams at Sites 1–3 and interviewed the team members at Sites 2–3. We also received individual audio recordings of each of Site 1 participants, and audio recordings of all distributed sidebars.

The task of space mission design involves constant problem-solving. The design involves choosing a number of different parameters, e.g. trip time, weight, power type, as well as graphically designing the spacecraft configuration. Parameters begin with initial estimates and are constantly refined. The work is highly interdependent, e.g. the power engineer needs information from the mission design and instruments expert before she can calculate her values. The interdependencies in the design decisions leads to the high degree of interaction to negotiate values or discuss design tradeoffs.

Different levels of sensemaking in the team process: Reciprocity of perspectives

At Site 1, for the most part, all team members were familiar with each others' identities and were aware of their areas of expertise. More commonly, collocated team members shared perspectives in the design process. They were all concerned with keeping costs down, minimizing mass in the design, and assessing "technology readiness levels" to estimate the amount of research and development needed between that design session and the commencement of mission operations. Shared perspectives emerged in individual interaction for example when team members made guesses about the meanings and implications of numeric values in a spreadsheet, which were confirmed by another member indicating a shared understanding.

An essential aspect of design is the capacity to explore various scenarios for benefits and risks. When, in the course of such exploration, a feature is identified and its implications are immediately grasped by another, it indicates a reciprocal understanding of the situation. For example, a telecommunications hardware expert expressed concern about the effects of cold temperatures found in space on an instrument to penetrate the surface of a spatial body. The Science and Instrumentation experts immediately grasped that cold-induced brittleness was a risk that had not yet been identified in this design.

Other reciprocal perspectives reflect local norms and attitudes. One engineer at Site 1 told another that he was going to "pick on him," reflecting a local norm encouraging informed peer critique. In another situation, two team members at Site 1 sought advice from a non-Team member, also at Site 1, rather than seek assistance from a team member at a remote site.

Finally, shared perspectives were reflected in the common practice of conveying design information in a "shorthand" manner by referencing similar information from prior designs. For example, when one CDS engineer referred to a data system design as "Seeker," the other CDS expert immediately understood. Similarly, a shorthand reference to the "Cassini" mission for the schedule was then used by another person.

In contrast, the full, distributed design team experienced difficulty in establishing shared perspectives for the design process. Discrepancies, in both understanding and in the actual design parameters, occurred. Design decisions made at each site were often reported on the third day, and there was insufficient time to track down subsequent design decisions that had already been made with the discrepant values.

Thus, sensemaking was conducted differently within sites compared to across sites. The collocated team members exhibited behaviors that indicated that they shared common perspectives, especially with respect to the design process. In contrast, in the entire design team, many incidents occurred that pointed to a lack of common perspectives. These differences led to three consequences for the team, discussed next.

Sidebars as scripts: the "heart" of design work

In space mission design, much of the "heart" of design work occurs in smaller groups, or sidebars, where clarifications or design tradeoffs are discussed. A characteristic of the sidebars at Site 1 was their spontaneity. At any time, from one to five sidebars usually occurred at the Site. People continually monitored the environment, listening for keywords in the surrounding discussions that had relevance for them. When such a keyword was detected, the team member would spring up from their seat and join the sidebar. Importantly, nearly all sidebars were self-organized. Thus, it was expected that the patterned interaction of sidebars was the standard type of encounter in this collocated setting. Rarely did the facilitator organize a sidebar. Sidebars could range from a simple and quick question, such as for clarification or to seek specific information ("what is the temperature of Mars?") to a lengthy and complex design tradeoff discussion, such as how to reduce weight on the spacecraft.

In contrast, sidebars did not exist as standard types of encounters for the larger distributed design team. Sidebars were always delegated by facilitators who announced publicly over the VTC who would join them. All sidebars were held via teleconferencing. It was perfectly possible for any team member to initiate a sidebar across distance by asking the coordinator for a phone line and dialing the other site. Yet of the 24 distributed sidebars that occurred, only three were self-organized. The distributed sidebar interactions generally involved complex discussions of longer duration, generally around a single topic. The distributed team never used sidebar interaction to spontaneously clarify, seek information, or challenge a design value or assumption. This would have been advantageous for the design team, e.g. if the Power engineer in Team 1 clarified a value with the Power engineer in Team 2.

Thus, a pattern of behavior never emerged where distributed team members would spontaneously contact their colleagues across distance as the need arose. The coordination overhead may have prevented people from spontaneously engaging in distributed sidebars; it took an average of three minutes, 17 seconds to set up a distributed sidebar. In fact, no other generalized forms of distributed interaction, or scripts across sites, were detected.

Discrepant methodologies and assumptions

When intersubjective meaning is achieved in a group, it follows that all group members understand terms and processes in the same way. During the design session, the different sites not only used different concepts and terms, but also unique methodologies and design processes. In three cases, the different sites used different methodologies for concepts that are standard in mission design, e.g. in computing contingent mass. They also used different terms for standard concepts, e.g. "trajectory". These cases revealed two things. First, though each site had developed a common understanding of its own terms, a lack of shared understanding existed across sites. Second, intersubjectivity depends on actors performing the work to maintain and develop shared perspectives. When attempts at establishing shared meanings were made by proposing hybrid terms, these were not adopted by the design team. The sites did not make the requisite effort to allow intersubjectivity to emerge in the entire team by committing to the decision. Though the design team was able to intellectually negotiate the common terms and methodologies, the design team did not have congruent perspectives established that would enable them to adopt the solutions.

Blind trust in technology

A third consequence that we observed is that misattributions occurred during the distributed design team interaction. Participants at the different sites developed a blind trust that the collaborative tools that they used to interact and share data across distance were "delivering" the information they intended. The actors behaved as though their distributed partners would perceive their behaviors and work practices in the same way that their local team members would.

Examples included "what I say is what you hear". There were 24 instances, spread approximately equally over Sites 1–3, where team members did not put in the requisite effort in public conversations to make themselves heard at the other sites. Team members at remote sites complained that the site who spoke recently could not be heard. The speaker either forgot to unmute the microphone, or spoke too far away from the VTC microphone to be heard remotely. Another example of blind trust is "what I see is what you see" when people at one site expected other sites to see the same displayed value on the networked spreadsheet. Still another example is falsely believing "what data I can access is what you can access" across sites. They expected that once values were entered into the spreadsheets, they were immediately propagated and accessible to the other remote sites which was not always the case.

While interaction is easy within a site, it was not clear to participants that they needed to invest extra effort to understand how the remote members' perceived their behaviors and work practices conveyed by technology. Also, for most people, they were using new and unfamiliar technologies and did not have the opportunity to develop appropriate expectations of the capabilities of the technology [1].

Discussion

In this paper we have investigated a new interaction order of large-scale group-to-group collaboration. We discovered that in the collocated sites, sensemaking tended to be mostly intersubjective, i.e. that people's perspectives were congruent and reciprocal. In the larger distributed design team, sensemaking was far less intersubjective. Sensemaking has different facets and we can interpret the differences between collocated and distributed groups by examining these facets. Table 1 shows more specifically how different components of sensemaking relate to the three consequences that we observed.

Communication breakdowns in the design team were triggers for developing intersubjectivity, i.e. for the team to move to a different level of sensemaking. An example of such a breakdown was when discrepant methodologies were discovered, as for contingent mass. The breakdown had the potential of being a catalyst for the design team to develop shared meanings. The team succeeded partially as new emergent terms did develop as a result of conversations, and were unique to the design team. Yet intersubjectivity was not actually constructed across distance, as the design team did not adopt the new terms. Each site reverted back to the use of their own terms, knowing that it was not accepted by the other sites. The compromise agreement for contingent mass that each site would apply their methodology for that part of the design they were responsible for is not a viable longterm solution. This agreement was also not adopted. Design is an iterative process and the use of discrepant methodologies may lead to incongruencies downstream in later stages of mission design. This poses a risk to the design (and mission).

Observed consequences	Components of sensemaking	Distributed design team	Collocated teams
Sidebars	Sensemaking as expected patterns of behavior	Only three self-organized sidebars; the rest are delegated and not spontaneous, but formal; coordination intensive; identities of partners not always known	Spontaneous joining of sidebars; monitoring sidebars; articulation as well as design sidebars; identities of partners mostly known
Adoption of terms	Sensemaking as commitment	Though common terms were negotiated and agreed upon, they were only temporarily used; not permanently adopted	Common language and guidelines were developed and used
Misattributions or "blind trust" in technology use	Sensemaking as expectation	Have not developed appropriate set of expected behaviors for technology use across distance; not aware when human use of technology breaks down	Breakdowns in human use of technology are usually visible

Table 1. Different components of sensemaking in the distributed design and collocated teams.

Intersubjectivity does not remain constant but vascillates between the intrasubjective and generic subjective states and must be maintained. The team process is a cycle of alignment and breakdown. Breakdowns can lead to the identification of points where common meanings can be established. When alignment occurs, intersubjectivity has the opportunity to emerge. The nonadoption of the common terms by the entire design team and misattributions were examples of how the design team transitioned away from intersubjectivity. If communication repair occurs, then it is a step towards intersubjectivity.

A major risk for large-scale scientific collaborations is when perspectives are not questioned. At local sites we observed many instances of spontaneous challenges to e.g., a design parameter or assumption. These occurred mostly in sidebar discussions, but also in large public discussions within the site. Debate and negotiation were the norm. In contrast, we rarely observed spontaneous challenges made by team members across distance. The facilitators sometimes questioned a perspective or a value, but the mission design would benefit more by having nonfacilitators, or experts in multiple specialties, introduce challenges. Distributed sidebars, where design tradeoffs were discussed, were mostly limited to formal discussions of predefined topics by team members assigned by the facilitators.

It was not our expectation that intersubjectivity or generic subjectivity would be achieved by the design team as it did not have much experience meeting together. Our goal in this paper was rather to examine the consequences of what happens when groups in large-scale collaborations experience *different levels* of intersubjectivity and practice different types of sensemaking. Such short-term interaction is not uncommon in large-scale ad-hoc collaborations such as when scientific teams discuss a problem using the Access Grid.

REFERENCES

[1] Olson, G. M. and Olson, J. S. Distance Matters, *Human-Computer Interaction*, *15,* 2/3, 2000, 139–178.

[2] Parkin, K. L., Sercel, J. C., Liu, M., and Thunnissen, D. P., ICEMakerT: An Excel-Based Environment for Collaborative Design, *IEEE Aerospace Conference,* (Big Sky, MT, March 8-15, 2003). IEEEAC Paper #1564. Updated Jan 25, 2003. http://monolith.caltech.edu/Papers/Parkin%20IEEE%20Paper%201564.pdf

[3] Schutz, A. *On Phenomenology and Social Relations: Selected Writings,* H R. Wagner (ed). University of Chicago Press, Chicago, IL, 1970.

[4] Weick, K. E. *Sensemaking in Organizations,m Sage Publications.* Thousand Oaks, CA, 1995.

[5] Wiley, N. (1988). The Micro-macro problem in social theory. *Sociological Theory*, vol. 6, 254-261.

Session: Computer-Supported Learning Environments

Reflective Design Practices in Human Computer Interaction and Software Engineering

Effie Lai-Chong Law
Eidgenössische Technische Hochschule Zürich
ETH Zentrum, TIK, Gloriastrasse 35, CH-8092 Zürich, Switzerland
law@tik.ee.ethz.ch

ABSTRACT
Three theories of reflection of Dewey, Vygotsky and Schön can presumably well inform the development of different aspects of HCI and software engineering. Upon reflecting on the related literature, we derive some boundary conditions for reflective design practice and formulate three questions, of which the understanding is enhanced through the three theoretical models of reflection.

INTRODUCTION
In the recent literature on design, be it of architectural constructions, software systems, or professional training programs, Donald Schön's theory of reflection [18,19] has frequently been referenced. In fact, Schön's theory is rooted in that of Dewey[1] and Vygotsky[27]. A common thread linking the social constructivist theories of these three scholars is that knowledge and actions are fundamentally social in origin, organization and use, and are situated in particular context. Presumably, Dewey's social pragmatic, Vygotsky's socio-linguistic and Schön's communicative views of reflection (see below) can well inform the development of different aspects of Human-Computer Interaction (HCI) and software engineering (SE). Specifically, some boundary conditions for reflective design practice can be derived.

In HCI and SE various design models have become popular in the last decade, including participatory design, situated design, scenario-based design, user-centered design, and evolutionary design. A basic tenet shared by these models is that design, as a form of creativity enabled by ample opportunities for reflection [11], is essentially a social practice – a core concept echoing the three views of reflection. Furthermore, the ever-increasing research interest and effort in these social constructivist approaches (cf. rationalist-cognitivist approach) to design [28] has paralleled the prolonged analyses on human-machine and work-technology relationships [3]. Both lines of inquiry can mutually influence each other. For instance, the supposition that there is an inherent asymmetry between human beings and machines in terms of their differential access to resources embedded in the social and material environment [21, 22] can inform the design of intelligent software agents [15]. Moreover, the social constructivist approaches to design imply that software engineers need to address a vast array of issues when designing a system. Reflective thinking is necessary to cope with the overwhelming demand. The concomitant question is: *What should designers reflect on*?

The recalcitrant gap between HCI and SE has lately drawn much attention and concern of professionals from both domains [9]. Issues pertaining to usability have been one of the starting points to coordinate the efforts of HCI specialists and software engineers. There exist joint endeavors on incorporating usability into software architecture, identifying usability patterns, constructing taxonomies of usability problems, and improving usability evaluation methods (UEMs). Since analyzing each of these aspects is beyond the scope of this position paper, we elaborate our view only on UEMs, of which several problems are basically design in nature, including design of usage scenarios, evaluation procedures and tools, and data analysis scheme. A number of UEMs have been criticized as not adequately rooted in a sound theoretical framework and rather pragmatic in nature. Since evaluation is essentially a reflective practice, we assume that theories of reflection can somehow enhance our understanding of UEMs. The concomitant question is with such an increase in knowledge: *How UEMs can be rendered more effective?*

Besides, the definition of usability is problematic. The core definition of usability concept as a set of measurements (i.e., ISO 9241) is too limited and too technical to explain phenomena and to support design and research activities when social and cultural aspects have to be dealt with [6]. Contextual approaches to usability have been put forward but not yet adequately explored. Similarly, the notion "cultural usability" has been addressed. It is a working hypothesis for a design practice that reaches beyond the functional interests of contemporary usability research and interface development by situating design in its wider socio-cultural contexts [25]. While the technical definition of usability is too narrow, its social counterpart can be too broad to manage. The concomitant question is: *What is the manageable scope of usability?* In the ensuing discussion, we first briefly delineate the three theoretical models of reflection and then derive some boundary conditions for reflective design. Next, we examine how they can improve our understanding of the three questions raised above.

THEORIES OF REFLECTION
Historically, the challenge of defining reflection has been entertained by scholars of different epochs. For Dewey, it is

a preferred form of thinking triggered by doubt and perplexity perceived in a situation, resulting in problem resolution in light of previous experiences. For Vygotsky, reflection is the transferal of argumentation from a social level to an internal one. For Schön, it is a dialogue of thinking and acting through which performance can be enhanced. In sum, the definition of reflection is beset by its temporal(anticipatory, contemporaneous, and retrospective) and developmental dimensions (ranging from technical to critical reflection).

Dewey's Social Pragmatic View of Reflection

According to Dewey[1], the role of reflection is to regulate the dialectic relationship between knowing and acting, and reflective thinking is a tool for problem resolution and operates through the progressive cycle of 'inquiry'. An inquiry is a teleological impetus for determining a course of actions to counteract instability of a situation. There exist two types of inquiry. Whereas a perceptual inquiry entails adapting to the affordances of a situation and results in ad-hoc actions, a reflective inquiry entails manipulating symbolic representations and leads to planned actions. Besides, Dewey's evolutionary point of view implies that reflective inquiry develops out of perceptual inquiry through persistent agent-world transactions. Dewey[2] emphasized the role of tools in the emergence of mind, especially language. In accord with Dewey's pragmatic social behaviorism, communication and action in a social setting can facilitate reflective thinking.

Dewey[1] postulated five phases of reflective thinking: problem recognition; enumeration of possibilities of new actions or beliefs; evaluation of the possibilities through consulting memory, questioning, or experimenting; revision of possibilities; decision-making on next appropriate actions. These phases, varying in duration with the type of inquiry, can overlap in time. He also specified three attitudes required for reflection: open-mindedness, absorbed interest and responsibility in facing consequences.

Vygotsky's Sociolinguistic View of Reflection

According to Vygotsky [27], reflection can be understood as self-regulation, which is acquired by a process that involves first experiencing "other-regulation" which occurs in the zone of proximal development where adult guidance or collaboration with more capable peers is available. Through this special mode of social interaction, the form and content of self-regulation are gradually transferred from the more competent partner and internalized by the learner. The Vygotskian views also stress that sociolinguistic experience is indispensable for the emergence of metacognition and that intersubjectivity is a primary means for knowledge construction. The corollary is that modeling and verbal communication (including self-verbalization) are strong facilitators for reflection.

Vygotsky also advocated the thesis that reflection plays a mediating role by transforming meaningful experiences into learning which leads to development. Vygotsky, like Dewey, regarded language as the most potent cultural tool in achieving convergence of meaning and co-construction of knowledge during social interactions. Based on Vygotsky's theory of dialectical relationship between the intra- and inter-psychological and transformation of one into another, high-order thinking like reflection is developed through consistent agent-world dynamic interactions.

Schön's Communicative View of Reflection

According to Schön [18,19], reflection-on-action and reflection-in-action as essential factors for the development of professional artistry, which refers to kinds of embodied skills practitioners demonstrate in problematic situations of practice. Whereas reflection-on-action refers to thinking back on the action already accomplished or pausing in the midst of an action to make a "stop-and-think" (i.e., offline), reflection-in-action occurs while a practice is being undertaken (i.e., online) and implies moment-by-moment "active experimentation". Besides, reflection-in-action is conceptually more complex, developmentally more mature, and functionally more significant than reflection-on-action. Based on his communicative views, Schön believed that the effectiveness of a practicum depends crucially on social interactions, especially reciprocally reflective dialogues between coach and student who have to maintain communication which eventually leads to convergence of the interpretations of the concepts in question.

Schön's model of reflective practice consists of four central components: perceiving an indeterminate zone of practice precipitated by instability of a specific situation; framing the problem in terms of the particulars of the situation, analyzing and criticizing such an initial problem framing; reframing the problem in light of the inquirer's repertoire of domain-specific knowledge and previous experiences; generating moves for future actions leading to the new coherence of the situation. This sequence of operations can be seen as an individual's attempt to converse with the situation in which he is embedded. Reflective conversation is a highly dynamic and dialectical cognitive enterprise. The inquirer shapes the situation, but in conversation with it, his idiosyncratic methods and appreciations are in turn shaped by the situation.

Implications of the Three Views of Reflection to Design

Based on the basic assumption that reflective thinking is requisite for design activities, we infer some boundary conditions for design from the three views of reflection.

- First, design is essentially a social practice and mediated by artifacts and tools socio-historically constructed, of which language is particularly important. Hence, collaborative working environments, where pluralistic and meaningful social discourse among stakeholders is supported, are conducive to design [12].
- Second, design entails a contextualized problem and a source of stimulation, which, according to Dewey and Vygotsky, can be described as dialectical transactions

between internal and external. Hence, to design systems with any integrity, it is imperative for designers to develop them in relation to specific settings of use [23] and to sustain ongoing interactions with the social and material environment, which can 'talk back' [18] to designers to propel the related works.

- Third, design is inherently evolutionary in nature, undergoing progressive and iterative steps (cf. Dewey's evolutionary view on perceptual and reflective inquiry; Vygotsky's notion of spiral cognitive development; Schon's "framing-reframing" cycle). Hence, design plans (cf. requirement specifications in SE) have to be flexible and adaptive so as to accommodate emergent needs. Plans can actually serve as a kind of resource to bridge the gap between knowing and acting [9,21]. In fact, evolutionary approaches to design have been advocated by some contemporary scholars [4, 14].
- Fourth, design is a highly dynamic mental activity that tends to overburden our cognitive load [24] - a problem closely related to the issue of intrinsic motivation. Hence, objects of reflection should not be too encompassing. Besides, well-articulated but negotiable goals, which are somehow compatible with institutional arrangements, need to be set, thereby increasing the designer's sense of ownership of the problem as well as his or her motivation. Besides, the attitudes of open-mindedness and responsibility have to be reinforced.
- Fifth, design anchors in a rich declarative and procedure knowledge base. This explains expert-novice qualitative and quantitative differences in design activities. Hence, to enrich the skills required, it is desirable to provide designers with just-in-time training or tutorial support. Among others, expert modeling seems to be a relatively promising training strategy.

PROBLEM RESOLUTIONS

In this section, the three question posed in the foregoing discussion will be examined. We point out that each of the questions touches upon a large scope of intricately related problems. While we cannot provide any conclusive answers, we aim to stimulate further reflective conversations in the community of practice and interest.

What Should Designers Reflect on?

Identifying appropriate objects of reflection is the foremost and crucial step leading to the personal and professional growth. We propose an expanding scope of reflection with

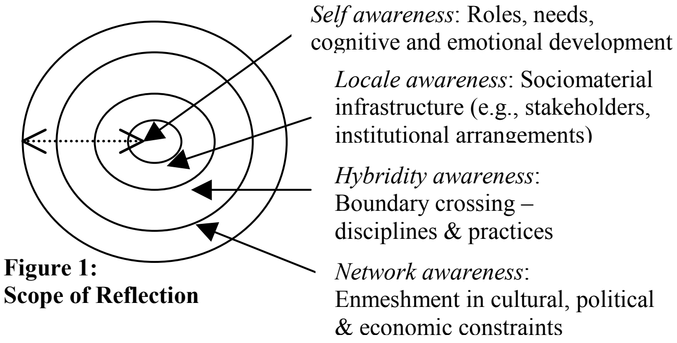

Self awareness: Roles, needs, cognitive and emotional development
Locale awareness: Sociomaterial infrastructure (e.g., stakeholders, institutional arrangements)
Hybridity awareness: Boundary crossing – disciplines & practices
Network awareness: Enmeshment in cultural, political & economic constraints

Figure 1: Scope of Reflection

four levels of awareness (Figure 1). Evolution of self [13] is the most significant function of reflection; consolidating a coherent self enables one to reach out to other levels. Locale is defined as a setting where design works get done. It is imperative for designers to be aware of what kinds of resources (e.g., expert guidance, reusable ideas in database) are accessible and what local constraints they must observe.

Hybridity implies our constant moving across disciplines and practices, leading to frequent shifts of perspectives [25] and even feelings of alienation and inadequacy [23]. Nonetheless, domains are not natural entities [20] and disciplinary boundaries can be seen as artifacts created to sustain the power and vested interest of their upholders [7]. The implication of this line of argument is that designers need to reflect on their roles in relation to elements of the socio-material infrastructure that constitutes technical systems. Besides, to optimize contributions of the workforce involved in a design project, the following factors are deemed necessary: reciprocal learning of complementary concepts, genuine respect for divergent views, high accountability, and ongoing dialogue facilitated by (partial) translation of the concepts in interest. It is noteworthy that the perimeters of the concentric rings presented in Figure 1 are "permeable" in the sense that the flows of information and knowledge among the four levels are basically possible, as illustrated by the double-arrow. With consistent practices of reflective activities entailed by the four levels of awareness, designers can develop an integrated view of the field where they are embedded.

How UEM can be Rendered More Effective?

Co-construction of knowledge is a paradigm commonly upheld by the three views of reflection. Collaborative discourse is congruent with reflective approaches to knowing because articulation to others helps one to share and clarify one's ideas. Mutual intelligibility of the concepts of interest can be attained through ongoing negotiations among conversational partners [21]. The implication is that the effectiveness of UEMs can be enhanced in a collaborative context. For empirical UEMs such as usability tests, in contrast to the 'standard' arrangement where single users work independently, there exist team usability tests where users in dyad or a small group co-discover usability problems while collaboratively performing given task scenarios on prototypes or operational products [5, 26]. For analytic UEMs, there exist collaborative usability inspections [10], where usability experts, representative users, developers, and graphic designers jointly identify usability problems in prototypes or models. However, whether these "social-based" UEMs are more cost-effective than their "individual-based" counterparts in detecting usability problems cannot yet be consistently confirmed by the empirical data. The key may lie in the techniques employed for extracting data on the first place (i.e., thinking aloud in usability tests; heuristics/principles selected) and in the methods adopted to compare the effectiveness of different UEMs. Nevertheless, we believe that collaborative usability evaluation is a promising approach worthy of closer investigation.

What is the Manageable Scope of Usability?

The three views of reflection are rooted in the socio-constructivist theories, which have challenged the basic assumptions underlying the rationalist-cognitivist tradition and dethroned its hegemony. Indeed, different types of phenomena entail different frameworks to make sense of them. Similarly, the narrow, technical definition of usability should be supplemented (not replaced) by a broader, social one. Another reason for the need of an alternative definition is the "ubiquitization" of human-machine interaction with concomitant increase in user heterogeneity and their needs. What they require from a product is more than effectiveness and efficiency. Hence, usability is measured more in qualitative rather than quantitative terms and more experiential rather than conceptual. With the shift from a rigid to a relatively fluid conceptualization of usability, we need to review existing UEMs. Specifically, we may have to ask users different questions concerning their emotional, aesthetical, ethical, attitudinal, and social values towards the usage of a product. One crucial point is that usability problems are relative to product and service goals. The challenge is how to map different UEMs to different goals. Such mappings may serve as general guidelines, and usability specialists need to adapt them to the particularities of an application context. We assume that the manageability of social-based usability can be optimized if there are well-coordinated collaborations among stakeholders and usability is addressed at the very beginning of a product design and sustained throughout the process.

Concluding Remark

We cannot provide any conclusive answers to the three questions we posed, partly due to the limited empirical data available. But they are significant issues that need to be addressed in the future research of HCI and SE. We remark that a reflective design practice implies critical sensibility to design [25]. Accordingly, we should be cognizant of the tacit assumptions underlying the discourses and usages of new technologies, and of the socio-historical background of existing cognitive tools (e.g., metaphors, taxonomies, templates) with which design artifacts are represented and constructed.

REFERENCES

1. Dewey, J. (1933/1986). How we think. In J.A. Boydston (Ed.), *John Dewey: The later works, 1925-1953, Vol. 8* (pp.105-352). Carbondale & Edwardsville: Southern Illinois University Press.
2. Dewey, J. (1925/1981). Experience and nature. In J.A. Boydston (Ed.), *John Dewey: The later works, 1925-1953, Vol. 1* (pp.1-326). Carbondale & Edwardsville: Southern Illinois University Press.
3. Greenbaum, J. & Kyng, M. (Eds.) (1991). *Design at work*. Hillsdale, NJ: Erlbaum.
4. Fischer, G., & Ostwald, J. (2002). Seeding, evolutionary growth, and reseeding. In T. Binder, J. Gregory, I. Wagner (Eds.), Proceedings of the Participatory Design Conference pp 135-143, Malmö University, Sweden.
5. Hackman, G. S. & Biers, D. W. (1992). Team usability testing: Are two heads better than one? In *Proceedings of the Human Factors Society 36th Annual Meeting*, 1205--1209. Santa Monica, CA: HFES.
6. Keinonen, T. (2001). From usability engineering to interaction design [online]. Access at: http://www.mlab.uiah.fi/culturalusability/papers/Keinonen_paper.html
7. Lave, J. (1988). Cognition in practice. Cambridge University Press.
8. Law, L.-C. (1998). A situated cognition view about the effects of planning and authorship on computer program debugging. *Behaviour & Information Technology, 17*(6), 325-337.
9. Law, L.-C. (2003). Bridging the HCI-SE gaps: Historical and epistemological perspectives. In M. Morten & J. Vanderdonckt (Eds.), Proceedings of the Workshop on Closing the Gaps (pp. 47-54), Zurich, 1-2 Sept 2003.
10. Lockwood, L.A.D., & Constantine, L.L. (1999). *Software for use*. New York: Addison-Wesley.
11. Loveless, A.M. (2002). *Literature Review in Creativity, New Technologies and Learning.* A report for NESTA Futurelab [online]. At: http://www.nestafuturelab.org/reviews/cr01.htm
12. Mamykina, L., Candy, L., & Edmonds, L. (2002). Collaborative creativity. *Communications of ACM, 45*(10), 96-99.
13. Mead, G.H.(1934). *Mind, self, and society*. Chicago: University of Chicago Press.
14. Mørch, A.I. (2003). Evolutionary growth and control in user tailorable systems. In N. Patel (Ed.), Adaptive Evolutionary Information Systems (pp. 30-58). Idea Group Publishing.
15. Nwana, H.S., & Ndumu, T. (1999). A perspective on software agents research. *The Knowledge Engineering Review,14* (2), 1-18.
16. Ohnemus, K. R. & Biers, D. W. (1993). Retrospective versus concurrent thinking-out-loud in usability testing. *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting*, 1127--1131. Santa Monica, CA.
17. Redmiles, D.F. (2002). Supporting the end users' views. Working Conference on Advanced Visual Interfaces (Trento, Italy), May 2002, pp. 34-42.
18. Schön, D. A. (1983). *The reflective practitioner*. New York: Basic Books
19. Schön, D. A.(1987). *Educating the reflective practitioner*. San Francisco: Jossey-Bass.
20. Simon, H.A. (1996). *The sciences of the artificial* (3rd Ed.). Cambridge, MA: MIT Press.
21. Suchman, L. (1987). *Plans and situated actions*. New York: Cambridge University Press.
22. Suchman, L. (2000). Located accountabilities in technology production [online]. Access at: http://www.comp.lancs.ac.uk/sociology/soci039ls.html
23. Suchman, L. (2003). Figuring service in discourses of ICT: The case of software agents. In E.H. Wynn, E.A. Whitley, M.D. Myers & J.I. DeCross (Eds.), *Global and organizational discourse about information technology, 33-43*. Boston: Kluwer.
24. Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction, 4*, 295-312.
25. Tarkka, M. (2001). Towards a critical design practice [online]. Access at: http://www.mlab.uiah.fi/culturalusability/introduction.html
26. Van Kesteren, I.E.H., Bekker, M.M., Vermeeren, A.P.O.S., Lloyd, P.A. (2003). Assessing usability evaluation methods on their effectiveness to elicit verbal comments from children subjects. In Proceedings of the 2003 Conference on Interaction Design and Children, Preston, England, pp. 41-49.
27. Vygotsky, L. S. (1978). Mind in Society. Cambridge, MA: Harvard University.
28. Winograd, T., & Flores, F. (1987). Understanding Computers and Cognition. New York: Addison-Wesley.

Position paper submitted to workshop on "Designing for Reflective Practitioners", CHI 2004, Vienna - Austria, April 25th.

Knowledge Building in Distributed Collaborative Learning: Organizing Information in Multiple Worlds

Anders Mørch
InterMedia
University of Oslo
Norway
+47 22840713
anders.morch@intermedia.uio.no

Karianne Omdahl
Department of Information Science
University of Bergen
Norway
+47 55558381
karianne.omdahl@uib.no

Sten Ludvigsen
InterMedia
University of Oslo
Norway
+47 22840712
sten.ludvigsen@intermedia.uio.no

ABSTRACT

In the CSCL (Computer Supported Collaborative Learning) community a recent topic of keen debate has been whether or not online discussion forums should be typed or not (i.e. information categorized according to predefined message types). We have analyzed findings from a field trial with Future Learning Environment (FLE) and we identified some problems with the system's knowledge building categories. We propose to integrate collaborative knowledge building with physical modeling (designing with materials) to get more mileage out of information categorization. This is stimulated by Donald Schön's bottom-up approach to information categorization, from design materials to repertoires of cases.

1. Design according to Schön

In a series of empirical studies of professionals in a range of domains Schön (1983) has shown that information categorization to a large extent is bottom up work rather than originating from readymade categories. This process starts from "materials of a situation" and in a good process of design results in new understanding realized as a "case" added to a existing repertoire of cases The notion of a *repertoire* is more fluid than a concept and constructed out of the local, often messy, situation a person finds himself in when solving a design problem, but in the end is linked with existing understanding so that it can be reused in future situations requiring similar problem solving. A repertoire is thus distinguished from a category set by being the result of a combination of bottom up (situation specific) sense making and top-down structuring of existing understanding. In his own words, analyzing an architect at work, Schön describes the design process as follows:

"When a practitioner makes sense of a situation he perceives to be unique, he sees it as something already present in his repertoire. To see this site as that one is not to subsume the first under a familiar category or rule. It is, rather, to see the unfamiliar, unique situation as both similar to and different from the familiar one, without at first being able to say similar or different with respect to what. The familiar situation functions as a precedent, or a metaphor (Schön 1983, p. 138).

The quote suggests categories (as flexible repertoire) should be allowed to evolve over time, stimulated and informed by a reciprocal relation of adaptation and situational "back talk". Adaptation occurs when categories are used locally and the back talk provides feedback to regulate the adaptation process so that it makes sense to the participants.

Even though students are not professionals in the sense just described they need to take part in similar processes to successfully learn. For example learners need to engage in a process of grounding, i.e. interaction necessary to establish a common ground to complete collaboration tasks (Baker et al., 1999, Koschmann & LeBaron, 2003). Physical modeling by manipulating domain-specific materials is one form of grounding appropriate for conceptual knowledge building. The following quote by Donald Schön is illuminating in this regard:

"the designer's moves tend, happily or unhappily, to produce consequences other than those intended. When this happens, the designer may take account of the unintended changes he has made in the situation by forming new appreciations and understanding and by making new moves (Schön, 1983, p. 79)."

Design according to this occurs on two levels. On the one hand, it is about "forming new appreciations and understanding," on the other it is about "making moves" in the domain. Moves with unintended consequences can serve as triggers for conceptual knowledge building by identifying new problems (framing of issues) that may require exploration and explanation before new moves can be made.

2. Conceptual Knowledge Building

CSCL focuses on technology in its role as mediator of activity within a collaborative setting of instruction and learning, learners and facilitators. It has inherited its intellectual legacy from theoretical schools in the social sciences, in particular sociology, anthropology, and communication (Stahl, 2002). Knowledge, from this perspective, is seen as a human construction elaborated through communication and collaboration with peers, mediated by social and cultural artifacts implying that learning and knowledge building first of all occur on inter-personal grounds within a community of learners before occurring on the intra-personal realm of the individual learner (Vygotsky, 1978).

A pedagogical model developed within this perspective is *Knowledge Building* (Scardemalia & Bereiter, 1994). Knowledge building entails that new knowledge is not simply assimilated with the help of a more knowledgeable person, but also jointly constructed through solving problems with peers by a process of building mutual understanding in some domain of inquiry. Knowledge building and its subsequent refinement *Progressive Inquiry* (Hakkarainen, Lipponen, & Järvelä 2002) have received considerable attention in the CSCL community. A reason for this is that the model fits well with the educational philosophy instituted by many schools in Canada and Scandinavia (problem-based learning), as well as elsewhere in the world. The basic idea is that students gain a deeper understanding of a knowledge domain by engaging in a research-like process in this domain by generating their own problems, proposing tentative hypotheses and searching for deepening knowledge collaboratively with peers.

FLE (Future Learning Environment) is an open-source learning environment (http://fle3.uiah.fi/) developed according to the Progressive inquiry model. It is an asynchronous web-based groupware for computer supported collaborative learning (Muukkonen, et al., 1999). It is designed to support collaboration in the form of a discussion forum with message categories (information types) named after the stages of the progressive inquiry model.

Students using FLE are required to choose a knowledge-building category each time they post a message to other students. Although the initial questions were articulate and easily entered into FLE, responding to them by selecting a new information type was more difficult. In an empirical study (Ludvigsen & Mørch, 2003; Mørch, Dolonen & Omdahl, 2003) we identified recurring problems with using the FLE categories (content/category mismatches). We also identified student strategies of resolving them, such as trial and error: referencing a subset (or the whole range) of the categories to see if any one of them could apply. This strategy of information categorization is partly supported by the system. However, the teacher would also on occasion tell the students what each of the categories meant.

3. A Proposal for Integrated Knowledge-Building Environments

When the categories of a groupware are inappropriate to a situation at hand it may be because the situation is unique. Rather than forcing a "best match" on top of the situation the category be expandable and adaptable to the situation. This may have the dual effect of engaging those with skills to create new categories as well those with difficulties using the existing category set.

Although information categorization can be remedied by making categorization structures more transparent (e.g. with the use of everyday terms) we do not want to water out categorization structure entirely, since semi-structured messages can be surprisingly useful as basis for computer support (Malone et al., 1987). Instead, we propose a combination of user-tailorable categories and domain-specific designer kits with computational design materials serving as electronic lenses transforming and connecting the local situation with the conceptual information space.

Many knowledge domains consist of domain-specific rules and building blocks that adhere to general principles that define broad conceptual spaces within which small-scale experiments can mark individual trajectories (e.g. mathematics, physics, biomedical engineering). These design elements or "domain distinctions" (Fischer et al., 1995) are not exploited in the current generation of knowledge building environments. On the contrary, the term knowledge building has become synonymous with manipulation of *conceptual artifacts* (Bereiter, 2002). Although the computer is well equipped to support conceptual artifacts as we have shown above, it is even better equipped to support modeling and simulation of physical phenomena, which we have tentatively dubbed "physical knowledge building" to complement conceptual knowledge building.

Modeling and simulation of physical phenomena is not foreign to designers of collaborative learning environments and has been acknowledged as being important for stimulating learning activity in many knowledge domains (e.g. Papert, 1991; Fischer et al., 1995; Roschelle et al., 1999). However, this approach has received little attention in the knowledge building community and few attempts have been made at building bridges across the two worlds from the other side. We start by making a first move and suggest that the following hypotheses should be implemented and empirically tested in the next generation of knowledge building environments:

- Integrated knowledge building environments are needed for full support of distributed collaborative learning
- Integrated knowledge building environments need computer support for conceptual and physical knowledge building within the same computational environment
- Physical knowledge building can be supported by domain specific designer kits
- Designer kits need to align with the established domain distinctions of a particular knowledge domain
- Designer kits will make it easier for physically active students to engage in knowledge building
- Designer kits can complement existing (conceptual) knowledge building environments and help to focus collaboration activity
- End-user tailorability and intelligent agents are two computational techniques that can help to link general information categories with domain-specific, situations of a designer kit
- Automatic (adaptive) classification by the computer suggesting categories on the basis of analysis of the current situation in the learning environment

4. Related Work

The following past (and contemporary) work and system building describe related initiatives, directly and indirectly.

Grace (Atwood et al, 1991) was an integrated learning environment for Cobol programming. The Grace environment consisted of a suite of tools for different aspects of programming. For example, the environment included an intelligent tutoring component, a Cobol construction kit, and a Cobol critic. The system was field-tested in the training center at corporate headquarter of a regional telephone company (NYNEX). It was also a single user environment and implemented on the Symbolics machine.

VKB (Virtual Knowledge Builder) (Shipman, et al., 2002) is a distributed, spatial hypertext system allowing multiple users at different sites to manipulate shared ideas. The system is domain-independent, and implemented to allow collective creation, editing and manipulating of free-form textual notes. VKB has a set of suggestion agents that can recognize certain semantic attributes and values of the notes and suggest various ways to classify and reorganize them. For example a type suggestion agent can analyzes the attributes and visual properties of a newly create (untyped) note and suggest a classification for it based on matching it with the existing set of typed (categorized) notes, and by doing so helping the users with grouping ideas into meaningful clusters.

Epsilon (Soller, 2001) is an intelligent facilitation agent that is integrated with a shared graphical editor for the domain of object-oriented analyses and design using OML (Object Modeling Language). Collaboration among students is scaffolded by everyday sentence-openers (such as "Do you know", "Please show me", "Let me explain it this way, "To justify", "To summarize", etc) modeled after speech act theory, but in a more user friendly way. Epsilon can observe a group's conversation and dynamically analyze individual contributions. For example, it can recognize events such as a student having failed to discuss his or her work with others. When it detects an opportunity to react, the agent might intervene by asking the group to explain the student's actions. If the students in return are not able to select the proper sentence openers for this type of utterance the agent might intervene and tell them about the role of explanation in group learning. Epsilon continues the top-down tradition of information categorization, but the categories are now easier to select because they are mixed with everyday terms.

References

[1] Atwood, M.E., Burns, B., Gray, W. D., Morch, A. I., Radlinski, E. R. and Turner, A. The Grace Integrated Learning Environment: A Progress Report. *Proceedings, the Fourth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems* (Koloa Hawaii, June 1991), ACM Press, 741-745.

[2] Baker, M., Hansen, T., Joiner, R. and Traum, D. The Role of Grounding in Collaborative Learning Tasks. In Dillenbourg, P. (ed.) *Collaborative Learning: Cognitive and Computational Approaches*. Amsterdam, Pergamon, 1999, 31-63.

[3] Bereiter, C. *Education and Mind in the Knowledge Age*. New Jersey. Lawrence Erlbaum, 2002.

[4] Fischer, G., Lindstaedt, S., Ostwald, J., Stolze, M, Sumner, T. and Zimmermann, B. From Domain Modeling to Collaborative Domain Construction. *Proceedings of Designing Interactive Systems* (Ann Arbor, 1995), 75-85.

[5] Hakkarainen, K., Lipponen, L., and Järvelä, S. Epistemology of Inquiry and Computer-Supported Collaborative Learning. In T. Koschmann, R. Hall, and N. Miyake (eds.). *CSCL 2: Carrying Forward the Conversation.* Lawrence Erlbaum, 2002, 129-156.

[6] Ludvigsen, S. and Mørch, A. Categorization in Knowledge Building: Task-specific Argumentation in a co-Located CSCL Environment. Proceedings of CSCL 2003, Bergen, Norway. Kluwer Academic Publishers, pp. 67-76.

[7] Koschmann, T. and LeBaron, C.D. Reconsidering Common Ground: Examining Clark's Contribution Theory in the OR. Proceedings ECSCW'03. Kluwer Academic, 81-98.

[8] Malone, T.W., Grant, K.R., Lai, K.Y., Rao, R., and Rosenblitt, D. Semi Structured Messages are Surprisingly Useful for Computer-Supported Coordination, *ACM Transactions on Office Information System* 5, 2 (1987), 115-131.

[9] Mørch A, Dolonen J, Omdahl K. Integrating Agents with an Open Source Learning Environment. In: Lee KT, Mitchell K, ed. *Proceedings of International Conference on Computers in Education 2003 (ICCE 2003),* Dec. 2-5, Hong Kong: AACE Press, 2003.

[10] Muukkonen H., Hakkarainen K., Lipponen L., Leinonen, T. Computer Support for Knowledge Building. *9th European Congress on Work and Organizational Psychology, Innovations for Work, Organization and Well-being* (Espoo-Helsinki, Finland, May 1999).

[11] Papert, S. Situating Constructionism. In Harel, I. and Papert, S. (eds.). *Constructionism*. Ablex Publishing, 1991, 1-12.

[12] Roschelle, J., DiGiano, C., Koutlis, M., Repenning, A., Jackiw, N., and Suthers, D. Developing Educational Software Components. *IEEE Computer* 32, 9 (1999), 50-58.

[13] Scardemalia, M. and Bereiter, C. Computer Support for Knowledge-Building Communities. *The Journal of the Learning Sciences* 3, (1994), 265-283.

[14] Schön, D.A. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.

[15] Shipman, F.M., Moore, J.M., Maloor, P., Hsieh, H., and Akkapeddi, R. Semantics Happen: Knowledge Building in Spatial Hypertext. *Proceedings Hypertext 2002* (College Park, June 2002), 25-34.

[16] Soller, A.L. Supporting Social Interaction in an Intelligent Collaborative Learning System. *International Journal of Artificial Intelligence in Education* 12, 1 (2001), 40-62.

[17] Stahl, G. (ed.). Computer Support for Collaborative Learning: Proceedings of CSCL 2002. Lawrence Erlbaum, 2002.

[18] Vygotsky, L. S. *Mind and Society*. Harvard University Press, 1978.

CycleTalk: Supporting Reflection in Design Scenarios with Negotiation Dialogue

Carolyn Penstein Rosé, Vincent Aleven
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA, 15216 USA
cprose,aleven@cs.cmu.edu

Cristen Torrey
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh PA, 15216 USA
ctorrey@andrew.cmu.edu

ABSTRACT

In this paper we discuss the motivation for a novel style of tutorial dialogue system that emphasizes reflection in a design context by engaging students in negotiation dialogues. Our current research focuses on the hypothesis that negotiation-style dialogue will lead to better learning than previous tutorial dialogue systems because (1) it motivates students to explain more in order to justify their thinking, and (2) it supports the students' meta-cognitive ability to ask themselves the right questions about the choices they make.

Author Keywords
Tutorial Dialogue, self-explanation, design

INTRODUCTION

Among the most important skills one can develop during one's education regardless of the chosen profession or trade is the ability to think critically and construct sound arguments. For example, these skills are foundational to effective conflict resolution, which is a basic facet of all business relations. Furthermore, widespread accessibility of information in recent years, such as through the internet, is empowering individuals to take the initiative to educate themselves about their legal rights and medical options. However, in order for people to benefit from this initiative, they must be prepared to evaluate, filter, and synthesize potentially conflicting information from a wide assortment of sources in order to be able to argue their own interpretation of this morass of information. The centrality of reflection, critical thinking, and argumentation is perhaps most clearly seen in the scientific arena. Understanding how science connects with the real world on a conceptual level involves building mental models, in other words arguments, that use scientific principles to explain why objects interact the way they do. On another level, critical thinking and argumentation are at the heart of the scientific method. Finally, argumentation comes into play when science is applied in an engineering scenario when design trade-offs are evaluated in the light of scientific understanding. Because of these reasons, we are developing the CycleTalk tutorial dialogue system that supports the development of critical thinking and argumentation skills by engaging students in negotiation dialogues in natural language to immerse students in scientific inquiry at these three key levels: (1) *understanding science* at a conceptual level, (2) *doing science* by forming and then testing predictions using a simulator, and (3) *using science* for evaluating design trade-offs. A key feature of our approach is to engage students in negotiation dialogues for the purpose of stimulating reflection and drawing out their reasoning along these three lines and encouraging them to clarify their own thinking.

MOTIVATION FROM PREVIOUS WORK

The important role of language in learning has been affirmed many times from many different angles and with respect to many different types of subject matter. Recent research on student self-explanations supports the view that when students explain their thinking out loud it enhances their learning [6,7,8,17]. A human tutoring study in the Basic Electricity and Electronics domain [19] revealed a trend for Socratic style tutoring dialogues to be more effective for learning than didactic style ones. A possible explanation for this result is that students learn more effectively when they are given the opportunity to reflect and discover knowledge for themselves in an active way [4,14,16]. Stevens, Collins, and Goldin (1979) report that the best teachers tend to use a Socratic tutoring style. A follow-up analysis of the BEE corpus [9] demonstrates a significant correlation between ratio of student words to tutor words and learning, underscoring the importance of encouraging students to talk more during tutorial dialogue. In support of this, an analysis of the WHY2 human tutoring corpus has demonstrated a significant correlation between average student turn length and learning [21]. In further support of the importance of students expressing themselves through language as part of their learning, Chi et al. (2001) demonstrate that students in a pure self-explanation condition performed no worse than students in a human tutoring condition.

These results have spawned an optimistic view about the potential for building highly effective tutorial dialogue systems, capable of combining the advantages of both individualized instruction and interaction in natural language. Significant progress has been made with respect to this research agenda. Many tutorial dialogue systems have been built and have been evaluated with students, often in realistic educational settings [1,18,10,13,3,11]. These formative evaluation studies demonstrate that state-of-the-art computational linguistics technology is sufficient for building tutorial dialogue systems that are robust enough to be put in the hands of students and to provide useful learning experiences to students. A number of these studies show that tutorial dialogue systems have advantages over instructional treatments that do not involve dialogue. At times, however the comparison results were inconclusive, demonstrating that the field is still young and that there is much room for growth.

An evaluation of the AutoTutor system, a tutorial dialogue system in the domain of computer literacy, showed an advantage over re-reading of the textbook of about 0.5 standard deviations [15]. The textbook re-reading condition itself was no better than a no-treatment control condition. Similarly, a recent evaluation of WHY-AutoTutor, a system based on the same architecture as the original AutoTutor but applied to the domain of qualitative physics, demonstrates a significant advantage of this system over a textbook reading control [12]. However, in a different experiment the learning results obtained with WHY-AutoTutor were no better than those in a control condition in which students read targeted "mini-lessons," short texts that covered the same content as that presented in the dialogue [11]. The mini-lesson condition is different from reading textbook text in that mini-lessons tend to be focused specifically on the knowledge and potential misconceptions involved in a specific exercise. It appears to be a high standard against which to compare. Even human tutors are not always more effective a mini-lesson control, although human tutors are significantly more effective than a mini-lesson control condition with students who have no prior background with the subject material [21].

An evaluation of Andes-Atlas, a tutorial dialogue system for the domain of physics, which leads student through directed lines of reasoning, implemented by means of Knowledge Construction Dialogues (KCDs), demonstrated a significant advantage of Andes-Atlas without a significant increase in time-on-task, compared to an otherwise equivalent version of Andes which provided hints rather than dialogues [18]. The results of this study are however somewhat difficult to interpret due to a very high dropout rate (57%).[VA1] While Atlas' KCDs were shown to be more effective than hints in this evaluation of Andes-Atlas, in a different experiment they were not more effective than mini-lessons of the same kind as were used in the evaluation of WHY-AutoTutor, mentioned above [20,23].

A third tutorial dialogue system, the Geometry Explanation Tutor, which is still under development, was evaluated in two classroom studies. As students solve geometry problems, the system helps them through a restricted form of dialogue to state general explanations for their problem-solving steps. In the two evaluation studies, this system was compared against a version that was the same in all respects, except that students explained their steps by means of a simple menu instead of in a dialogue. In the first study, the students who explained in a dialogue had higher learning gains than students who explained by means of a menu [1]. However, the detailed pattern of results was difficult to interpret, in terms of the underlying knowledge that the students may have acquired, rendering the results somewhat inconclusive. In the second classroom study, carried out in a different school with better-prepared students, there was little difference between the two conditions [2].). The inconclusive result is likely to be due the fact that the students already had significant geometry knowledge.

Thus, tremendous progress has been made in the tutorial dialogue community in the past few years. Tutorial dialogue systems have been shown to lead to improved learning, compared to such as controls as textbook reading. At the same time, we know of no studies that have demonstrated conclusively that tutorial dialogue systems provide more effective or efficient instruction than some of the alternatives to which they have been compared, including an otherwise equivalent targeted "mini-lesson" based approach [11,20,23] and a "2nd-generation" intelligent tutoring system with simple support for self-explanation [1]. However, the situation sketched here does present a challenge. How does one develop a tutorial dialogue system that is more effective than the ones developed so far, especially where many of the systems built so far have a solid basis in empirical studies of human tutors and/or results in the cognitive science literature?

CYCLETALK

Our current research focuses on the hypothesis that negotiation-style dialogue will lead to better learning because (1) it motivates students to explain more in order to justify their thinking, and (2) it supports the students' meta-cognitive ability to ask themselves the right questions about the choices they make. Furthermore, we hypothesize that a more effective tutorial dialogue system would move beyond engaging students in understanding science into actually doing science and using science. In order to test that hypothesis, we are developing a novel style of tutorial dialogue system that pushes beyond the limitations of current tutorial dialogue technology by engaging students in negotiation dialogues in a design context. Specifically, we propose to develop CycleTalk, a tutorial dialogue system that builds on an existing "articulate simulator" in the field of thermodynamics. Building upon this foundation, the CycleTalk tutorial dialogue system will engage students in dialogues in which they negotiate the pros and cons of alternative designs for thermodynamic cycles, such as those that form the foundation for steam power plants or refrigerators.

Thus, CycleTalk will support students in understanding science by engaging them in discussions about how principles of thermodynamics play out in simulations of thermodynamic cycles. It will support them in actually doing science, by encouraging students to construct and defend predictions about how changes to example cycles would affect the cycle's efficiency or effectiveness and then testing those predictions using the simulator. Furthermore, it will engage students in using science by asking them to apply their understanding of thermodynamics to make and defend design decisions.

Design skills are essential and yet are difficult for students to acquire. Beyond understanding thermodynamics concepts and how and why individual factors can affect the efficiency of a cycle, design requires students to weigh and balance alternative choices in order to accomplish a particular purpose. Furthermore, design requires not only a theoretical understanding of the underlying science concepts but also a practical knowledge of how these concepts are manifest in the real world under non-ideal circumstances. Because of the intense demands that design places on students, we hypothesize that design problems will provide the ultimate environment in which students will be stimulated to construct knowledge actively for themselves.

For example, for each of the thermodynamic cycles that students explore with CycleTalk, the system will engage them in a discussion of the pros and cons of one or more alternative designs, such as changing the temperature or pressure within (certain) components of the cycle, adding components (e.g., a condensor, feedwater heater, re-heater), or changing the working fluid. Typically, the effect of such changes on thermodynamic efficiency will be considered, but the effect on other desirable characteristics (such as low cost of building the cycle or maintaining it) may be considered as well. As discussed, CycleTalk will employ a negotiation style of dialogue to encourage exploration of the space of alternative design choices.

Tutor: How do you think you could improve the efficiency of a simple Rankine cycle?

Student: Wouldn't it be possible to introduce a second turbine in series with the first one since the steam leaving the original turbine is still hotter than what enters the Boiler?

Tutor: A second turbine would improve the efficiency, but what about the possibility of some steam condensing on the blades of the second turbine as it continues to cool? Can you see a problem with this?

Student: No, I am not sure what you are getting at.

Tutor: What happens when steel continues to come into contact with water?

Student: it could rust?

Tutor: very good. Condensed water on the blades of the turbine would damage the blades. Can you think of a way to avoid this?

Student: A reheating unit could heat up the steam coming out of the first turbine so that it is not likely to condense onto the blades of the second turbine.

Tutor: Good idea! That solves the condensation problem, but what is the downside?

Student: Reheating requires an additional input of energy.

Tutor: True, so we should test the extent to which we would come out ahead of the game by doing this. But before we do that, can you think of any other disadvantages? Higher efficiency saves money, but does this come for free?

Student: No, the reheating unit costs money - between that and the additional turbine - not to mention additional maintenance. I wonder if it would be worth it in the long run...

This example illustrates that in CycleTalk, the point of the negotiation is to teach students to ask themselves the right questions, considering general issues such as efficiency, maintainability, durability, cost of parts, time, etc.

ACKNOWLEDGMENTS
This project is supported by ONR Cognitive and Neural Sciences Division , Grant number N000140410107. The authors would like to thank Ken Forbus for providing the CyclePad simulator to build on as a foundation for this work.

REFERENCES

1. Aleven V., Koedinger, K. R., & Popescu, O. (2003). A Tutorial Dialogue System to Support Self-Explanation: Evaluation and Open Questions. *Proceedings of the 11th International Conference on Artificial Intelligence in Education, AI-ED 2003.*
2. Aleven, V., Popescu, O., Ogan, A., & Koedinger, K. R (2003). A Formative Classroom Evaluation of a Tutorial Dialogue System that Supports Self-Explanation. *Supplemental Proceedings of the 11th International Conference on Artificial Intelligence in Education, AI-ED 2003. Volume VI.*
3. Ashley, K. D., Desai, R., & Levine, J. M. (2002). Teaching Case-Based Argumentation Concepts Using

Dialectic Arguments vs. Didactic Explanations. In S. A. Cerri, G. Gouardères, & F. Paraguaçu (Eds.), *Proceedings of Sixth International Conference on Intelligent Tutoring Systems, ITS 2002,* 585-595. Berlin: Springer Verlag.

4. Brown, A. L., & Kane, M. J. (1988). Preschool children can learn to transfer: Learning to learn and learning from example. *Cognitive Psychology, 20,* 493-523.
5. Chi., M. T. H., Siler, S., Jeong, H., Yamauchi, T., Hausmann, R. (2001). Learning from human tutoring. Cognitive Science, 25(4), 471-533.
6. Chi, M. T. H. (2000). Self-Explaining Expository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models. In R. Glaser (Ed.), *Advances in Instructional Psychology,* (pp. 161-237). Mahwah, NJ: Erlbaum.
7. Chi, M. T. H., de Leeuw, N., Chiu, M. H., LaVancher, C., (1994). Eliciting self-explanations improves understanding. Cognitive Science, 18:3, 439-477.
8. Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., and Glaser, R., (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:2, 145-182.
9. Core, M. G., Moore, J. D., Zinn, C., (2003) The Role of Initiative in Tutorial Dialogue*, Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.
10. Graesser, A. C., Bowers, C. A., Hacker, D.J., & Person, N. K. (1998). An anatomy of naturalistic tutoring. In K. Hogan & M. Pressley (Eds.), *Scaffolding of instruction.* Brookline Books.
11. Graesser, A., VanLehn, K., the TRG, & the NLT (2002). Why2 Report: Evaluation of Why/Atlas, Why/AutoTutor, and Accomplished Human Tutors on Learning Gains for Qualitative Physics Problems and Explanations, LRDC Tech Report, University of Pittsburgh.
12. Graesser, A. C., Jackson, G. T., Mathews, E. C., Mitchell, H. H., Olney, A., Ventura, M., Chipman, P., Franceschetti, D., Hu, X., Louwerse, M. M., Person, N. K., and the Tutoring Research Group, (2003). Why/AutoTutor: A Test of Learning Gains from a Physics Tutor with Natural Language Dialog. *Proceedings of the Cognitive Science Society.*
13. Heffernan, N. T., & Koedinger, K. R. (2002) An intelligent tutoring system incorporating a model of an experienced human tutor. In S. A. Cerri, G. Gouardères, & F. Paraguaçu (Eds.), *Proceedings of the Sixth International Conference on Intelligent Tutoring Systems, ITS 2002* (pp. 596-607). Berlin: Springer Verlag.
14. Lovett, M. C. (1992). Learning by Problem Solving versus by Examples: The Benefits of Generating and Receiving Information. In *Proceedings of the Fourteenth Annual Meeting of the Cognitive Science Society* (pp. 956-961). Hillsdale, NJ: Erlbaum.
15. Person, N., Bautista, L., Graesser, A., Mathews, E., & The Tutoring Research Group (2001). In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future, Proceedings of AI-ED 2001* (pp. 286-293). Amsterdam, IOS Press.
16. Pressley, M., Wood, E., Woloshyn, V. E., Martin, V., King, A., Menke, D. (1992). Encouraging mindful use of prior knowledge: Attempting to construct explanatory answers facilitates learning. Educational Psychologist, 27, 91-109.
17. Renkl, A. (2002). Learning from worked-out examples: Instructional explanations supplement self-explanations. *Learning & Instruction,* 12, 529-556.
18. Rosé, C. P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001a). Interactive Conceptual Tutoring in Atlas-Andes, In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future, Proceedings of AI-ED 2001* (pp. 256-266). Amsterdam, IOS Press.
19. Rosé, C. P., Moore, J. D., VanLehn, K., Allbritton, D., (2001b). A Comparative Evaluation of Socratic versus Didactic Tutoring, *Proceedings of the Cognitive Sciences Society.*
20. Rosé, C. P., Bhembe, D., Siler, S., Srivastava, R., & VanLehn, K. (2003a). Exploring the Effectiveness of Knowledge Construction Dialogues, *Proceedings of the 11th International Conference on Artificial Intelligence in Education, AI-ED 2003.*
21. Rosé, C. P., VanLehn, K., & the Natural Language Tutoring Group (2003b). Is Human Tutoring Always More Effective Than Reading: Implications for Tutorial Dialogue Systems, *Proceedings of the AIED 2003 Workshop on Tutorial Dialogue Systems: With a View Towards the Classroom.*
22. Rosé, C. P., Bhembe, D., Siler, S., Srivastava, R., VanLehn, K. (2003d). The Role of Why Questions in Effective Human Tutoring, *Proceedings of the 11th International Conference on Artificial Intelligence in Education, AI-ED 2003.*
23. Siler, S., Rosé, C. P., Frost, T., VanLehn, K. & Koehler, P. (2002). Evaluating Knowledge Construction Dialogues (KCDs) versus minilessons within Andes2 and alone, ITS Workshop on Empirical Methods for Tutorial Dialogue Systems, San Sebastian, Spain.
24. Stevens, A. L., Collins, A., Goldin, S. E., (1979) Misconceptions in Student's Understanding, International *Journal of Man-Machine Studies*, 11, 145-156.

Designing for, with, and by Reflective Practitioners

Finn Kensing
The IT University of Copenhagen
kensing@itu.dk

Donald Schön's thinking has influenced my research, teaching, and consulting since I first met him now more than twenty years ago. I was then a young researcher having just finished reading "The Reflective Practitioner". I traveled to Oslo to interview him during his visit there. He refused to let me interview him, but suggested instead to conduct an experiment through which I would learn about his approach. He arranged for me to interview three of his hosts about their ideas for new computer support to run their institution. He recorded the interview and stopped four times to conduct his now well known debriefing sessions.

It was the strongest learning experienced I ever had. He helped me become aware of the assumptions and pre-constructed figures of thought that guided my interviewing and thus my evolving local theory of the institution and the three people's perception of its computer needs. Ten years later we had him over to conduct a PhD course based on his thinking and the responses we got from many of the participants were as enthusiastic as my own.

My teaching and especially my supervision of projects owe much to Schön's line of thinking. Paying attention to the evolving categories and local theories developed by the students and assisting them in revealing and making active use of these is indeed a powerful teaching device.

Further, the idea of organizing activities through which practitioners are prompted to reflect on their own and others behavior as well as on their own and others conceptual constructs were instrumental in my coming up with the Prompted Reflection technique for understanding complex work (Kensing, 1998).

Last, but not least, when designing and disseminating the MUST method – a method for professional IT design (Kensing, Simonsen and Bødker, 1998a; Bødker, Kensing and Simonsen, 2004) – a Schönian approach was very helpful in many ways. When working as designers in more or less participatory projects we were designing *for* and *with* reflective practitioners (Kensing, Simonsen, and Bødker, 1998b; Bødker and Kensing, 1994; Simonsen and Kensing, 1997). When engaged in coaching IT designers using our method we were dealing with design *by* reflective practitioners (Kensing, 1999; Bødker, Kensing, and Simonsen, 2002).

Currently I am engaged in setting up a laboratory for the study and development of innovative design competence at the IT University of Copenhagen. The lab will be a 70 m2 room equipped with video and a repertoire of gradually evolving tangible design materials. We plan to conduct the first session in the lab when we move to our new building May 2004.

Bødker, K. and F. Kensing (1994): Design in an Organizational Context - an Experiment. In Scandinavian Journal of Information Systems, vol. 6, no 1, 1994.

Bødker, K., F. Kensing and J. Simonsen (2002): Changing Work Practices in Design. In Dittrich, Y. et al.: Social Thinking - Software Practice. MIT Press, 2002.

Bødker, K., F. Kensing and J. Simonsen (2004): Participatory IT Design. Designing for Business and Workplace Realities. MIT Press 2004.

Kensing, F. (1998): Prompted Reflections - a technique for understanding complex work. In ACM interactions, vol. 5, no. 1, 1998.

Kensing, F., J. Simonsen and K. Bødker (1998): Participatory Design at a Radio Station. In Computer Supported Cooperative Work - The Journal of Collaborative Computing, vol. 7, no 3-4, 1998.

Kensing, F., J. Simonsen and K. Bødker (1998): MUST - a Method for Participatory Design. In Human-Computer Interaction, vol. 13, no 2, 1998.

Kensing, F (1999): Method Design and Dissemination. In J. Pries-Heje et al (eds): Proceedings of The Seventh European Conference on Information Systems, Copenhagen, Denmark, 1999.

Simonsen, J. and F. Kensing (1997): Using Ethnography in Contextual Design. In Communications of the ACM, vol. 40, no. 7, July 1997.

The Reflective Information Systems Practitioner Approach as a Research and Learning Expedient

A Position Paper by
Ari Heiskanen, University of Oulu
(Ari.Heiskanen@Oulu.Fi)

In this position paper the author relates his experiences of combining practical information systems (IS) development work and scholarly research to other research and learning approaches. The overall framework, reflective information systems practice (RISP), is a development of Donald Schön's (1983, 1987) notion of the reflective practitioner. The specific research problems are drawn from the development history of administrative systems of Helsinki University from the early 1980's up to the present. The idea of RISP as a research approach grew gradually during the dissertation work of the author (Heiskanen 1994, 1995). The research began in 1987 as a positivistic inquiry to the implementation process of a new student record system of the University. In the early 1980's the author was a senior analyst developing the software and later a project leader for the decentralisation of the system functions to the departments of the University. During the process he also became the Chief Information Systems Officer of the University. Gradually during the late 1980's and the early 1990's the positivistic approach was replaced with a more hermeneutic or phenomenological view. Later the scope of research was enlarged to other fields, like in personnel and economic administration (Heiskanen and Assinen 2003; Heiskanen and Newman 1998; Heiskanen, Newman and Similä 2000), and cooperation between several universities when developing a common student record system (Heiskanen, Newman and Saarinen 1998).

Reflection is the practice of periodically stepping back to ponder on the actions of oneself and others in one's immediate environment (Raelin 2001; Seibert 1995). The object of reflection may be in three areas. First, content reflection is about how a practical problem was solved. Second, process reflection examines the procedures and sequence of the events. Third, premise reflection questions the presuppositions attending to the problem. The timing of reflection may be anticipatory, contemporaneous, or retrospective. Originally, Schön (1983, p. 163) characterised the work of design as a reflective conversation with the situation where the practitioner functions as an agent and an experimenter who is at the same time also a target or part of this experiment. He coined the term "reflection-in-action" to describe this.

Our RISP aims at instrumental organisational learning: how to successfully develop information systems for the University community. The main audience for learning are the managers, project leaders, and systems analysts of the University. Organisational learning involves a process that enables the acquisition of, access to, and revision of organisational memory, thereby providing direction to organisational action (Robey, Boudreau and Rose 2000).

The learning in RISP typically consists of consecutive cycles. Each cycle begins with a reflective comprehension of the situation that demands the action of the practitioner. Actions taken produce results that we call in the Schönian (Schön 1983) style organisational back-talk, indicating that the results of the action may be different from the planned ones. Back-talk leads to reflection, which, in turn, is a predecessor of new actions. We have illustrated our framing by presenting the histories of University systems development in a graphical format (Heiskanen 1995, Heiskanen and Assinen 2003; see an example in Figure 1). Our interpretation of the history is based on the interplay between issues and events, problems, and action strategies. An issue or an event describes an occurrence that needs a reaction. The problem defines our comprehension of the situation. The strategy defines the way the problematic situation is solved.

Many large information systems evolve through generations. The time taken may be several decades (e.g. Lasher et. al 1991; Short and Venkatraman 1992; Mason et al. 1997). In these long processes the learning cycles are also long. In our case, the development of the student records system contained four learning cycles during the years 1981-1993 (Heiskanen 1995), and the data warehouse development process 1990 – 2002 contained third learning cycles (Heiskanen and Assinen 2003).

As a research and learning expedient, RISP can be related to and compared with several approaches. First, as the practitioner stays within her organisation for an extended period of time, she is like an ethnographer in this respect (Heiskanen and Newman 1997). Second, as the practitioner is supposed to act in a meaningful way, she is like an action researcher (Heiskanen and Newman, forthcoming; cf. also Coghlan and Brannick 2002). Third, one aim of the reflective practitioner is organisational and individual learning; therefore this approach can meaningfully be compared (Heiskanen and Assinen 2003) to action learning (e.g. Revans 1980) and action science (Argyris et. al 1987). Fourth, as the reflective practitioner is an actor in the development history of the ISs of her organisation, she in a way is also a historian (cf. Mason et. al 1997). Table 1 contains a brief presentation of how to position these approaches with each other.

Research approach	**Key idea**
RISP	A versatile approach for anticipatory, contemporary or retrospective reflections and interpretations over work-life situations by a (single) practitioner, targeted for individual and organisational learning.
Ethnography	A participatory but typically non-obstructive way of research in which the researcher is long and deeply involved with the daily work life of the organisation under investigation.
Action research	A theoretically informed intervention approach, typically led by a researcher who has a client in the target organisation.
Action learning	A personnel development approach, used in group settings, that seeks to apply and generate theory from real work situations.
Action science	An intervention approach to help participants increase their effectiveness in social situations through heightened awareness of the assumptions behind their actions and interactions.
IS historian	An outsider researcher aims to tell a convincing story, often for learning purposes, based on documents and other sources describing the flow of events.

Table 1. Positioning RISP.

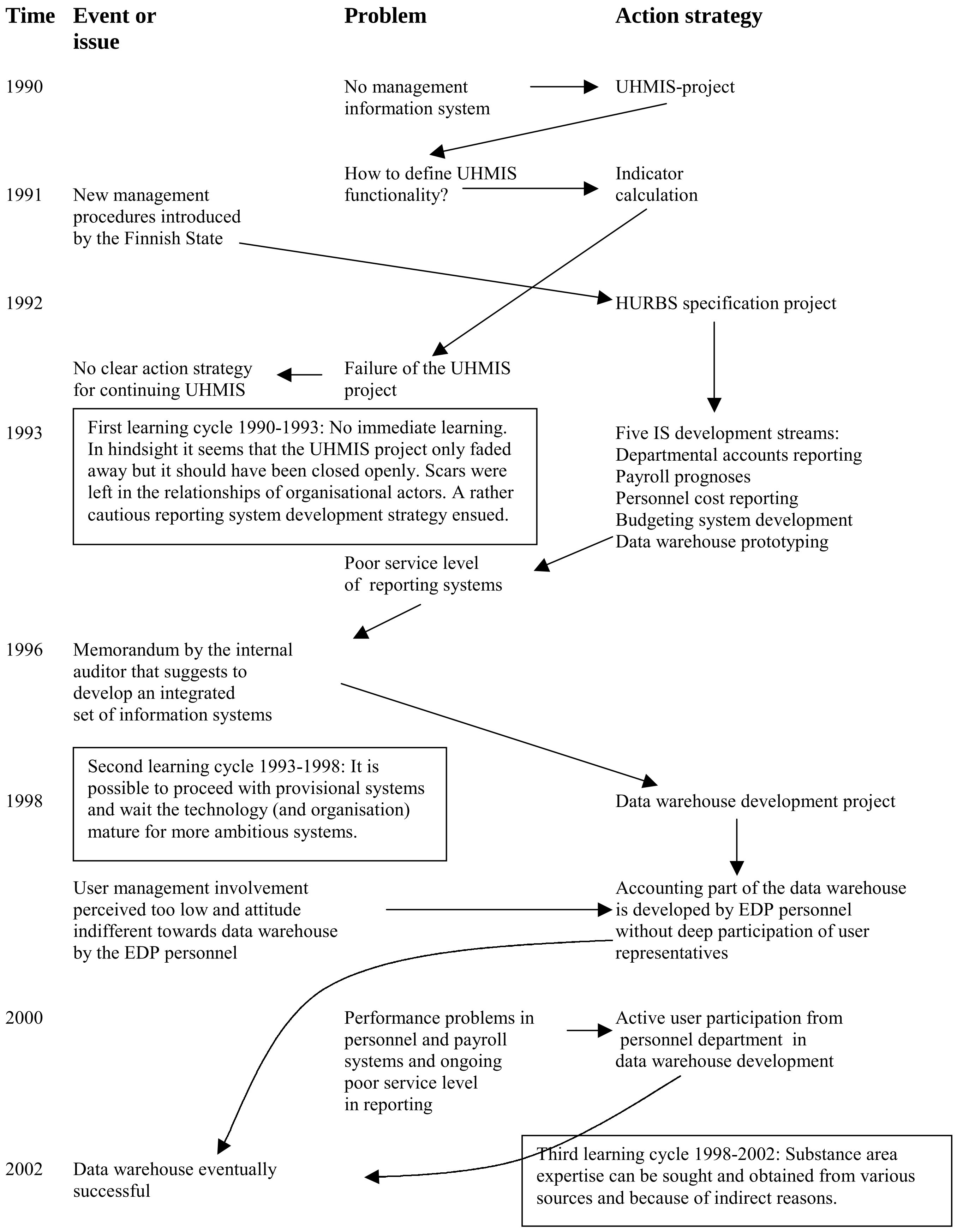

Time
Event or issue
Problem
Action strategy
1990
No management information system
UHMIS-project
How to define UHMIS functionality?
Indicator calculation
1991
New management procedures introduced by the Finnish State
1992
HURBS specification project
No clear action strategy for continuing UHMIS
Failure of the UHMIS project
1993
First learning cycle 1990-1993: No immediate learning. In hindsight it seems that the UHMIS project only faded away but it should have been closed openly. Scars were left in the relationships of organisational actors. A rather cautious reporting system development strategy ensued.
Five IS development streams:
Departmental accounts reporting
Payroll prognoses
Personnel cost reporting
Budgeting system development
Data warehouse prototyping
Poor service level of reporting systems
1996
Memorandum by the internal auditor that suggests to develop an integrated set of information systems
Second learning cycle 1993-1998: It is possible to proceed with provisional systems and wait the technology (and organisation) mature for more ambitious systems.
1998
Data warehouse development project
User management involvement perceived too low and attitude indifferent towards data warehouse by the EDP personnel
Accounting part of the data warehouse is developed by EDP personnel without deep participation of user representatives
2000
Performance problems in personnel and payroll systems and ongoing poor service level in reporting
Active user participation from personnel department in data warehouse development
2002
Data warehouse eventually successful
Third learning cycle 1998-2002: Substance area expertise can be sought and obtained from various sources and because of indirect reasons.

Figure 1. The Learning cycles in reporting systems development (Heiskanen and Assinen 2003; UHMIS is University of Helsinki Management IS, HURBS is a Reporting and Budgeting System.)

REFERENCES

Argyris, C., Putnam, R. and Smith, D. (1987). *Action Science.* San Francisco: Jossey Bass.

Coghlan, D. and Brannick T. (2002) *Doing Action Research in Your Own Organization.* London: Sage Publications.

Heiskanen, A. (1994). *Issues and Factors Affecting the Success and Failure of a Student Record System Development Process, a Longitudinal Investigation Base on Reflection-in-Action.* Doctoral Dissertation at the University of Tampere. Helsinki: the University of Helsinki.

Heiskanen, A. (1995). "Reflecting over a Practice, Framing Issues for Scholar Understanding." *Information Technology and People*, Vol. 8, pp. 3-18.

Heiskanen, A., Assinen, P. (2003). ”Learning Cycles, Organisational Back Talk, and the Persistence of Theories in Use: Lessons of Information Systems Development in a University Administration Context.” *Knowledge and Process Management*, Vol. 10, No 3, pp. 183-193.

Heiskanen, A. and Newman, M. (1997). ”Bridging the Gap Between Information Systems Research and Practice: The Reflective Practitioner as a Researcher”. *Proceedings of the Eighteenth International Conference on Information Systems,* Kumar, K. & DeGross, J.I. (Eds.), Atlanta, Georgia, December 15-17, pp. 121 - 131.

Heiskanen, A. and Newman M. (1998). ”The Dynamics of IS Procurement, Case Study of a Budgeting and Financial Reporting System”. *Proceedings of the 6th European Conference on Information Systems*, Baets, W. R. J. (Ed.), Aix-En-Provence, 3-6.6.1998, pp. 839-852.

Heiskanen, A. and Newman, M. (forthcoming). “The Reflective Information Systems Practitioner as a Researcher”

Heiskanen, A., Newman, M. and Saarinen, V. (1998). ”Innovations in Fiefdoms: Developing a Common Student Record System in Six Finnish Universities.” *Proceedings of the IFIP WG 8.2 and 8.6 Joint Working Conference on Information Systems,* Larsen, T. J., Levine, L. and DeGross, J. I. (Eds.), pp. 455- 469.

Heiskanen, A., Newman, M. and Similä, J. (2000).”The Social Dynamics of Software Development”. *Accounting, Management and Information Technologies*, Vol. 10, No 1, pp. 1-32.

Lasher, D. R., Ives, B., and Jarvenpaa, S. L. (1991). “USAA-IBM Partnerships in Information Technology: Managing the Image Project.” *MIS Quarterly,* Vol. 15, No. 4, pp. 551-565.

Mason, R. O., McKenney, J. L., and Copeland, D. G. (1997). “Developing an Historical Tradition in MIS Research.” *MIS Quarterly,* September, pp. 257-278.

Raelin, J. A. (2001). “Public Reflection as the Basis of Learning”, *Management Learning,* Vol. 32, No. 1, pp. 11-30.

Revans R. (1980). *Action Learning.* London: Blond & Briggs.

Robey D, Boudreau M-L, Rose G M. (2000). Information technology and organizational learning: a review and assessment of research. *Accounting, Management and Information Technologies*, Vol. 10, No 2, pp. 125-155.

Schön, D. (1983). *The Reflective Practitioner, How Professionals Think in Action.* New York: Basic Books.

Schön, D. (1987). *Educating the Reflective Practitioner, Toward a New Design for Teaching and Learning in the Professions*, San Francisco: Jossey-Bass.

Seibert, K. W. (1995). “Reflection-in-Action: Tools for Cultivating On-the-Job learning Conditions.” *Organizational Dynamics*, pp. 54-65.

Short, J. E. and Venkatraman, N. (1992). “Beyond Business Process Redesign: Redefining Baxter’s Business Network.” *Sloan Management Review,* pp. 7-21.

REFERENCES

Argyris, C., Putnam, R. and Smith, D. (1987). *Action Science.* San Francisco: Jossey Bass.

Coghlan, D. and Brannick T. (2002) *Doing Action Research in Your Own Organization.* London: Sage Publications.

Heiskanen, A. (1994). *Issues and Factors Affecting the Success and Failure of a Student Record System Development Process, a Longitudinal Investigation Base on Reflection-in-Action.* Doctoral Dissertation at the University of Tampere. Helsinki: the University of Helsinki.

Heiskanen, A. (1995). "Reflecting over a Practice, Framing Issues for Scholar Understanding." *Information Technology and People,* Vol. 8, pp. 3-18.

Heiskanen, A., Assinen, P. (2003). ”Learning Cycles, Organisational Back Talk, and the Persistence of Theories in Use: Lessons of Information Systems Development in a University Administration Context.” *Knowledge and Process Management*, Vol. 10, No 3, pp. 183-193.

Heiskanen, A. and Newman, M. (1997). ”Bridging the Gap Between Information Systems Research and Practice: The Reflective Practitioner as a Researcher”. *Proceedings of the Eighteenth International Conference on Information Systems,* Kumar, K. & DeGross, J.I. (Eds.), Atlanta, Georgia, December 15-17, pp. 121 - 131.

Heiskanen, A. and Newman M. (1998). ”The Dynamics of IS Procurement, Case Study of a Budgeting and Financial Reporting System”. *Proceedings of the 6th European Conference on Information Systems*, Baets, W. R. J. (Ed.), Aix-En-Provence, 3-6.6.1998, pp. 839-852.

Heiskanen, A. and Newman, M. (forthcoming). “The Reflective Information Systems Practitioner as a Researcher”

Heiskanen, A., Newman, M. and Saarinen, V. (1998). ”Innovations in Fiefdoms: Developing a Common Student Record System in Six Finnish Universities.” *Proceedings of the IFIP WG 8.2 and 8.6 Joint Working Conference on Information Systems,* Larsen, T. J., Levine, L. and DeGross, J. I. (Eds.), pp. 455- 469.

Heiskanen, A., Newman, M. and Similä, J. (2000).”The Social Dynamics of Software Development”. *Accounting, Management and Information Technologies*, Vol. 10, No 1, pp. 1-32.

Lasher, D. R., Ives, B., and Jarvenpaa, S. L. (1991). “USAA-IBM Partnerships in Information Technology: Managing the Image Project.” *MIS Quarterly,* Vol. 15, No. 4, pp. 551-565.

Mason, R. O., McKenney, J. L., and Copeland, D. G. (1997). “Developing an Historical Tradition in MIS Research.” *MIS Quarterly,* September, pp. 257-278.

Raelin, J. A. (2001). “Public Reflection as the Basis of Learning”, *Management Learning,* Vol. 32, No. 1, pp. 11-30.

Revans R. (1980). *Action Learning.* London: Blond & Briggs.

Robey D, Boudreau M-L, Rose G M. (2000). Information technology and organizational learning: a review and assessment of research. *Accounting, Management and Information Technologies*, Vol. 10, No 2, pp. 125-155.

Schön, D. (1983). *The Reflective Practitioner, How Professionals Think in Action.* New York: Basic Books.

Schön, D. (1987). *Educating the Reflective Practitioner, Toward a New Design for Teaching and Learning in the Professions*, San Francisco: Jossey-Bass.

Seibert, K. W. (1995). “Reflection-in-Action: Tools for Cultivating On-the-Job learning Conditions.” *Organizational Dynamics*, pp. 54-65.

Short, J. E. and Venkatraman, N. (1992). “Beyond Business Process Redesign: Redefining Baxter’s Business Network.” *Sloan Management Review,* pp. 7-21.

Session: Integrating Problem Framing and Problem Solving

Corson Associates

The Non-denominational Design Studio : Lessons for the Proficiency of Organizations

CHI Workshop Submission – 1/25/04

Design : the intentional transformation of an existing situation into a preferred situation.

From this definition of design, we (as individuals and as organizations) understand most of our daily life activities -- both professional and personal -- to be related to design. If we care about the quality of our life, we desire these daily efforts -- our intentional transformation of the situations we encounter -- to be as effective as possible. We want to accomplish the most possible with the resources available. When this involves familiar situations, we are comfortable. We find familiar problems and we solve them by applying familiar tools. If we are accomplished and creative, our solutions are very good.

The real world, however, tends to present itself not as familiar well-formed problems, but rather as messy, indeterminate situations. If we respond to these invariably unique situations by trying to see in them only conventionally familiar problems, our efforts will prove either lacking or completely useless. We all have found ourselves in this position. We have gone back and started over, we have re-doubled our efforts, we have checked our math again and again -- and the answer is still wrong. Regardless of how creative we are as problem solvers, we fail. This is inevitable. In our haste to find a familiar answer, we have "named" -- and are trying to solve -- the wrong problem.

What we learn from such failures -- if we desire to learn from them -- is that true creativity is based not in the solving of the problem, but in the naming of the problem we will apply ourselves to solving. The recognition of this distinction leaves most of us uneasy. Although we have excelled at solving known problems, we have had little experience with -- and even less encouragement for -- reflecting on whether we are solving the "right" problem. While we can see that to be the best designers possible we must develop this ability, we have no solid idea how to proceed to do so.

We need a new set of principles and perspectives to guide us in exploring this larger context - this realm of messy, indeterminate situations. We need a new set of principles and perspectives that are "non-denominational" -- that are broader than those of any specific discipline, that are owned by no one discipline and which can belong to and be understood and mastered by each and every one of us.

I have worked for many years in a wide range of cross-disciplinary design collaborations -- in the built environment, in curriculum development, in public policy, in the delivery of services and in the augmentation of organizational capabilities. These collaborations have always been conducted with intelligent, enthusiastic people who are eager to share their expertise. In nearly every experience, the dynamic of the working process has tended toward quickly deconstructing the "messy, indeterminate situation" into comfortably known problems. These processes are driven by the participants's eagerness to contribute their expertise. They begin, in effect, with the solution and then edit the situation to yield problems that fit their tools and techniques. This is not a path to systemic effectiveness, nor is it a path to the elegant expenditure of resources.

While I hold graduate degrees in both engineering sciences and architecture, my most important role in these collaborations has invariably been that of generalist: translator, facilitator and integrator. In this role, I have struggled to find a comprehensive and effective way to check this headlong rush to the "known" and redirect the collective enthusiasm and ability into a conscious and fruitful exploration of the "unknown."

I have become intrigued with how organizations -- both permanent and temporary -- sense and explore the messy, indeterminate situations in which they exist and how they extract from those situations the problems they choose to confront.

My work has become the development of a broad and integrated range of perspectives and principles that can be understood to underlie and precede traditional higher-order discipline-based design. These perspectives and principles are necessarily "non-denominational" -- that is, they are not restricted to or claimed by, nor solely comprehensible to any specific group or discipline. For that reason, I call them and the practice derived from them "non-denominational design." It is a practice accessible to each and every one of us and to each and every organization in which we participate.

I have found non-denominational design to be an effective foundation for robust collaborative practice. Its strength and effectiveness are supported from complementary sources: one internal: reflective practice, and the other external: Universal Design. The former, through reflection-in-action, fosters the continuous broadening of the designers's appreciation of both the situation with which they are struggling and the nature of their process for engaging in that struggle. The latter -- also a process of continuous awareness and interaction -- brings to the designers's assistance the interests, perspectives and experiences of the world external to their own.

I work with organizations to develop their non-denominational design capabilities. This effort begins with the creation of an in-house "non-denominational design studio" -- a space that enables collective exploration of systemic relationships and change. This design studio exists simultaneously as a physical, a virtual, an intellectual and a cultural space. It is a shared set of perspectives on the critical importance of designing robust design processes and is a shared awareness of behaviors that are congruent and incongruent with the effective pursuit of good design. Most importantly, and most problematically for its initiation, it is a space free of the roles and responsibilities and the hierarchies and accountabilities of the day-to-day "implementation organization." It is a set of activities that function laterally across the traditionally vertical process of conventional organizations. Through its purpose being "doing the right thing," the non-denominational design studio complements and significantly strengthens the conventional process-- whose primary purpose is "doing the thing right."

From this foundation, these organizations are able to undertake productive –- and otherwise unimaginable -- explorations of critical, but previously undetected or consciously avoided, “messy, indeterminate situations.”

I have explored cross-disciplinary collaborative design, based in these practices of the non-denominational design studio, with undergraduate and graduate students from many disciplines, NASA personnel, and public and corporate organizations and their leaders. Most have found the approach to be highly appropriate, highly engaging and highly beneficial.

Application of Collaborative Conflict in System Design: The Case of the New Millennium Program

Mark Bergman
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3425 USA
mbergman@uci.edu

Gloria Mark
Department of Informatics
University of California, Irvine
Irvine, CA 92697-3425 USA
gmark@uci.edu

HETEROGENEOUS SYSTEM DESIGN PROBLEM

Most organizations do not develop systems from scratch. Instead, to deal with economic constraints, organization demands, and customer needs, they often: a) identify possible problems that represent customer demand [10, 16], b) consider which possible technologies are useful to address these problems, e.g. commercial off-the-shelf software (COTS) [8] or open source software [12], c) select which problems to address, and d) implement a project to solve these problems. The initial choices for problem selection, in turn, involve a process of balancing and negotiating requirements from multiple sources [11, 13, 14, 17].

In some views, the hardest part of project design is identifying the problem to be addressed [9, 10]. As March (1994) [16] describes, there is rarely only one clear problem to choose to address. Indeed, problems, and their respective technology choices, co-exist in competition with one another. A problem-technology choice can be viewed as a prospective project for possible funding by an organization's principals. Principals are those who have the power and resources to authorize and fund a project [4]. Each problem-technology set represents a different (yet sometimes overlapping) group of stakeholders. Each stakeholder group has its own set of requirements that underlie their project choice. This set may overlap with other stakeholder groups' sets, but usually not with all of them. Each project represents one or more choices of technologies. Based on the requirements of all of the stakeholders, there are hundreds of different project possibilities. How does an organization choose which project to do?

Bergman and Mark (2002) [5] conducted an empirical field study to examine in detail the issues faced by practitioners in forming and stabilizing requirements during project selection and the procedures they created to overcome them. In this paper, we examine how the practitioners applied collaborative conflict as part of the process by which they selected system projects.

THE NEW MILLENNIUM PROGRAM (NMP)

The New Millennium Program (NMP) was started in 1994. It is located within the Jet Propulsion Laboratory (JPL). The Jet Propulsion Laboratory has been in existence for over 40 years. It had been involved in the design and development of technologies used in nearly all of the NASA (National Aeronautics and Space Administration) outer space and Earth based missions during that time, including landing on the moon and the Mars rover.

The main mission of the NMP is to perform space flight validation of new technologies [15]. It was created to address a problem of the lack of new technology utilization in space science missions. The primary reason science missions need new technologies is to reduce mission cost, allow a measurement, or enable a new function or capability. However, new technology is considered too risky for space use, and hence off-limits to science missions. By performing space flight validation on new technologies, these technologies become available for use in future space missions.

We studied the "NMP Formulation" process. This process deals with how the NMP selects the technologies for space flight validation. We found the formulation process to be in fact a *project selection process*. There are thousands of possible technologies that need space flight validation, hundreds of which are considered important by NASA directors and science mission technologists at any one time. The technologies tend to cluster into sets of related functionality, such as propulsion, communications, sensors and control systems. Each new technology was viewed by the NMP as a possible project choice. The NMP selection process has been developed and evolved over the last nine years to address this issue of project-technology choice.

COLLABORATIVE CONFLICT AS SYSTEM DESIGN SENSEMAKING

The general results of the study are presented in a variety of papers from Bergman and Mark [5-7] and Bergman and Buehler [1-3]. In this discussion, we focus on how the NMP applied reflective methodology during initial system design.

The NMP customers are called NASA theme technologists. They represent different ongoing space programs, such as studying the sun-earth connection, planets and other heavenly bodies. They represent a variety of diverse missions that want to use new technologies. Specifically they want the new technologies for their capabilities and, when possible, reduced costs. The missions they represent and work on have very diverse needs. All of these needs cannot be satisfied with a single or small set of technologies. There are ongoing needs for hundreds of different technologies. In addition, the relative needs of the different theme technologists are organizationally equal. Hence, there is no easy answer as to what technology to choose. Indeed, the NMP found they could not apply quantitative decision support methods due to the complexity of the problem [1]. They had to create and implement a repeatable method of selecting technologies that all of the stakeholders would support or the NMP program would fail.

The observed NMP selection process is quite complex. It is performed in two distinct phases: Pre-phase-A and Phase A. Pre-phase A focuses on determining what the NMP customers want, what new technologies are being developed and which of these technologies can address a subset of the customer requirements. Phase A is used to first gather external (to the NMP) technology and project proposals, review and rate them, and then select which proposals move on to implementation.

In both Pre-phase A and Phase A, we observed the application of collaborative conflict as a key methodology used in the selection process. We define collaborative conflict as 1) isolating specific, reoccurring predictable conflicts that occur during the selection process and 2) letting those either most directly affected by the conflicts or impartial to them engage the conflicts in a procedurally bounded manner. It is the application of preplanned, procedurally bounded, same-time conflict analyses in order to determine hidden or "tacit" information about technologies and their related project plans. We observed this information if difficult to obtain because is tends to represent the negatives (i.e. downsides) of a technology or project proposal. This is the information proposers do not include since it would very likely compromise their chance of the proposal being accepted.

Applied Collaborative Conflict Analysis

The NMP broke down and isolated repeatably observed conflicts in their processes. The main conflicts that needed to be addressed in the NMP selection were: customer demands as expressed through requirements; technology selection during a) technology concept areas (TCA) development b) external technology proposal review and selection, and c) project review and selection; application of the NMP filters to the technologies in a TCA; and authority level (organizational, project, technical) disagreements over which TCAs (and their technologies) to support.

In Pre-Phase A, the NMP focused first on the framing technical issues: determining customer requirements and finding available technologies. With this information, the NMP technologists pulled formed viable system designs to address common sets of requirements. They call the resulting early system designs technology concept areas (TCAs). At the end of Pre-phase A, the NMP selects which TCAs are going to be supported for an open technologies call.

Phase A begins with the technology call. It contains the requirements of the various TCAs that are being funded for the current NMP selection cycle. Any United States base supplier of technology (industry, government, university) can submit a technology proposal to the call. This is similar to a call for papers to a conference. Phase A uses review panels to rate and rank each proposal, first technical, and then project proposals. It ends with a final project selection and hence, selection of the technologies included in the project.

We now discuss how the NMP applied collaborative conflict in addressing their issues. We focus on there specific applications of collaborative conflict: 1) theme technologists and their requirements, 2) NMP technologists and the NMP filters, and 3) the technology and project proposal review panels.

Theme Technologists – The NMP technologists gave their customers, i.e. the theme technologists, proposed TCAs to examine. Each theme technologist was given a week to review and initially rate each TCA. The ratings were based on how well or poorly an individual TCA fit their perceived needs. After that time, the theme technologists were brought together in a meeting to discuss their ratings with one another.

These meetings were procedurally bounded by time, activity and outcome. The meetings were only one to two hours long,. They met once a week until they finished the task of considering all of the TCAs. They used a 1-3 rating system (3 being highest). They added a "silver bullet" to a specific TCA indicating each technologist's highest importance. They also stayed confined to the technologies presented to them, although suggestions about other technologies were allowed. Discussion was quick, detailed and precise. The whole task never took more than 3 meetings.

During these meetings, the theme technologists learned about each other's ratings and enquired as to each rating's rationale. They made sense of each other's positions and in so doing, improved their own sense of their ratings. This is seen by their working together to change ratings on various TCAs based on newly discovered mutual interests or lack thereof. The theme technologists used their a priori positional conflicts over requirements and technologies to discover more about each other positions and the technologies themselves. They applied and clarified their own (technical, economic and political) requirements as they worked through the various different technologies. In so doing, they addressed their own as well as the groups' technical, economic and political positions, which are represented in the final ratings. Furthermore, some of the feedback from this process was used to modify the TCAs to make them better fit the theme technologists' needs.

NMP Technologists – Another application of collaborative conflict was performed by the NMP technologists themselves. Each NMP technologist is an engineering expert within a specific domain. These domains include sensors, propulsion, communications, software control systems, and so forth. They oversee the technologies and proposals that fit their area of expertise. Over the years, they have learned to trust the expertise of the other technologists when discussing technologies that are not in their engineering domain.

The NMP has a set of "filters," i.e. ongoing requirements that are used to determine which technologies are appropriate for space flight test validation [1]. The NMP technologists apply these filters to each proposed technology to determine whether it can be consider by the NMP from selection. These filters bound the discussion about the technologies, i.e. a determination about how well each technology met or failed these filters.

These discussions occur in formal meetings or informal gatherings (especially over lunch) of the NMP technologists. A technologist presents his or her rationale to the colleagues and then they intensely engage the issue. The presenter describes how a technology they are considering should be accepted by the NMP, often by comparing the technology with similar technologies selected by NMP in the past. If there is not a general quick consensus (which we rarely observed happen), the other NMP technologists would point out specific instances in which the technology fails one or more filters. The most observed filters discussed were a) did the technology have to be space, or could it tested on earth, b) was the technology mature enough in its development to merit consideration and c) why this technology deserved consideration over other competing technologies. These discussions can last for days or weeks, especially on those technologies that may be testable on earth.

By considering each TCA and its component technologies against the NMP filters, the representing technologist must reaffirm their views as to why this technology should be tested. These discussions usually uncover new underlying issues about a technology that was not originally considered. These issues lend further insight to the strengths and weaknesses of the TCA system designs. Hence, the NMP filter discussions produced a better understanding of the TCAs (and the technologies therein) by the NMP technologists individually and as a whole. It helped to produce NMP team consensus on which technologies pass the filters as well as fostered better system designs.

Review Panels – In Phase A, the NMP forms independent review panels to rate and rank the technology proposals. At this point in the selection process, there has been a general call for technology proposals, based on the requirements set forth for each selected (in Pre-Phase A) TCA. Those proposals that survive the initial technology review process are invited to submit project proposals. These are also reviewed and ranked by another independent panel of space system design experts.

There are many procedural boundaries for the review panels. First, the review panels are made up of technology and system engineering experts. They are selected to cover the different engineering areas defined by the TCAs in the call for technologies. They cannot be part of any active proposals. Next, each proposal is first only considered against other proposals in the same technical domain, i.e. addressing the same TCA. The best of these are selected for the project proposal stage. The, the project proposals are compared at the level of full NASA missions, i.e. at a project level. Technical specifics are only discussed at this level if they have a direct impact on project execution and outcome. In addition, there are specific rating and ranking forms for each technology the panelists need to fill out. This standardizes the comparison results per panel.

The technology panels consider first how well each proposal meets the stated (TCA) requirements. Those that fail this are dropped from consideration. They then discuss the relative weaknesses of each proposal. They look for unwanted, yet unavoidable issues with the technologies. Examples include high cost, uncertain support, excessive mass, power, size, chemical or biological hazards, insufficient shielding to work in space, and so forth. We call these *negative requirements*. In other words, what stakeholders want, need, or desire to constrain are positive requirements, while that which they do not want, need, or cannot or overly constrain are negative requirements. No technology is perfect. Each has positive (wanted, desired, need) and negative capabilities and constraints.

Upon reflection, we assert the panels mainly exist to allow the managed, bounded conflict of engineering expert opinions on each technology to be expressed a rapid, focused manner. There was a desire to learn as much about a technology as is possible without seeing the technology first hand. This is based on a combination of examining what is written in the proposals in conjunction with the history of the technology and the reputation of the business team producing it. All of this was taken into consideration during the panel discussions. Individual panelists often shared insight on the history of a specific technology and its project team, which gave added depth to a proposal.

The outcome of these panel discussions was deeper and richer insights into the positives and negatives of each proposal. The panelists were able to compare and contrast proposals to find even more strengths and weaknesses. We found that proposals tend to focus on the positive, i.e. how the met stated TCA requirements and their technical capabilities within constraints. The panels tended to balance these proposals by identifying the negative requirements and undesired technical attributes and constraints. Altogether, collaborative conflict fostered an improvement in the information accuracy and dept of each technology and project proposal.

CONCLUSIONS

Proponents present the positive view of their technologies, requirements and project specifications. Via conflict, an informed opposition uncovers the negative aspects of the technologies, requirements, and projects. During planned, procedurally bounded collaborative conflict, the proponents and opponents are afforded a space to work together to defend the strengths and uncover weaknesses of their positions. This is performed in an attempt to better understand all of the issues that are part of initial system design.

Collaborative conflict in system design is similar to other proceedings, for instance legal trials, journal peer review, and quality assurance. Key similarities are a) the existence of known or expected conflicts (during a process), although the specific nature of the conflicts is not known in advance, b) predefined rules and procedures by which these conflicts are addressed, c) an expectation that bringing in opposing view will produce hidden and tacit “truths” about the situation, d) an acceptance of the outcomes by the involved parties, and e) only those parties with direct interest or organizationally endowed power are part of the process. The main difference is that the outcomes of early system design are far from concrete. Whereas a trial’s outcome is (reasonably) clear, systems design can change as the development process progresses. Much of the activity in initial design is making sense of the choices available and the implications of each choice. Also, the early rules and procedures of conflict engagement are as not well defined.

The NMP incorporates collaborative conflict in many aspects of its selection process. This indicates that it is a useful tool for reflective design practice. Yet, the challenge is to determine in general where collaborative conflict is useful throughout the system design and diffusion process. We propose that further research needs to be done to determine and understand how these apply in other system design situations, such distributed software systems, information systems, collaborative work systems, and alike.

REFERENCES

1. Bergman, M. and Buehler, M.G., Addressing an Inadequacy in Quantitative Analysis: Examining NMP Technology Selection. in *IEEE Aerospace 2003*, (Big Sky, MT, 2003), IEEE press.

2. Bergman, M. and Buehler, M.G., Analysis of the New Millennium Program (NMP) Flight Validation Process Using PTAM. in *Aerospace 2002*, (Big Sky, MT, 2002), IEEE Press.

3. Bergman, M. and Buehler, M.G., Applying the Authority-Activity Model to the New Millennium Program's Technology Selection Cycle. in *IEEE Aerospace 2004*, (Big Sky, MT, 2004), IEEE Press.

4. Bergman, M., King, J.L. and Lyytinen, K. Large Scale Requirements Analysis Revisited: The need for Understanding the Political Ecology of Requirements Engineering. *Requirements Engineering Journal*, *7* (3). 152-171.

5. Bergman, M. and Mark, G., Exploring the Relationship between Project Selection and Requirements Analysis: An Empirical Study of the New Millennium Program. in *RE 2002*, (Essen, Germany, 2002), IEEE Press, 247-254.

6. Bergman, M. and Mark, G., *In Situ* Requirements Analysis: A Deeper Examination of the Relationship between Requirements Formation and Project Selection. in *RE 2003*, (Monterey, CA, 2003), IEEE Press.

7. Bergman, M. and Mark, G., Technology Choice as a First Step in Design: The Interplay of Procedural and Sensemaking Processes. in *DIS 2002*, (London, UK, 2002), ACM Press, 224-234.

8. Boehm, B. Requirements that Handle IKIWISI, COTS, and Rapid Change. *IEEE Computer*, *33* (7).

9. Brooks, F.P. *The mythical man-month: essays on software engineering*. Addison-Wesley Pub. Co., Reading, Mass., 1995.

10. Carroll, J.M. *Making use: scenario-based design of human-computer interactions*. MIT Press, Cambridge, Mass., 2000.

11. Davis, A.M. *Software requirements: objects, functions, and states*. PTR Prentice Hall, Englewood Cliffs, N.J., 1993.

12. DiBona, C., Ockman, S. and Stone, M. *Open sources: voices from the open source revolution*. O'Reilly, Beijing ; Sebastopol, CA, 1999.

13. Karlsson, J. and Ryan, K. A Cost-Value Approach for Prioritizing Requirements. *IEEE Software*, *14* (5). 67-74.

14. Lamsweerde, A.v., Darimont, R. and Letier, E. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, *24* (11). 908-926.

15. Li, F.K. New Millennium Program Plan, NASA, JPL, Pasadena, CA, 2000.

16. March, J.G. and Heath, C. *A primer on decision making: how decisions happen*. Free Press, New York, 1994.

17. Mylopoulos, J., Chung, L., Liao, S., Wang, H. and Yu, E. Exploring Alternatives during Requirements Analysis. *IEEE Software*, *18* (1). 92-96.

Reflective Practitioners and Unselfconscious Cultures of Design

Gerhard Fischer
University of Colorado, Center for LifeLong Learning and Design (L3D)
Department of Computer Science, 430 UCB
Boulder, CO 80309-0430 – USA
gerhard@cs.colorado.edu

INTRODUCTION

The Center for LifeLong Learning & Design at the University of Colorado in Boulder [L3D, 2004] has focused its research over the last two decades on *conceptual frameworks* and *system building* efforts characterized by the following global objectives:

- not building expert systems, but *systems for experts*;
- supporting reflective practitioners by *increasing the back-talk* of the design artifacts;
- putting owners of problems in charge by supporting *human problem domain interaction*;
- creating open, evolvable systems facilitated by *meta-design* and the *seeding, evolutionary growth, reseeding process model*; and
- supporting *social creativity* among reflective design communities.

My workshop contribution will try to put these efforts into perspective, assess where our research efforts are today, and analyze specifically the mutually defining roles of reflective practitioners in unselfconscious cultures of design.

A CONCEPTUAL FRAMEWORK FOR DESIGN

The Nature of Design Problems. The primary challenge for designers is how to make sense out of *"situations that are puzzling, troubling, and uncertain"* [Schön, 1983]. Design requires reflective practitioners. Simon's description of a painter provides an example of design as a conversation with the materials of the situation: "*in oil painting every new spot of pigment laid on the canvas creates some kind of pattern that provides a continuing source of new ideas to the painter. The painting process is a process of cyclical interaction between the painter and canvas in which current goals lead to new applications of paint, while the gradually changing pattern suggests new goals*." [Simon, 1996].

Integration of Problem Framing and Problem Solving. Design problems are not analyzed in one step and then enacted in the next. The process of problem framing and problem solving has to be intertwined, and therefore the role of designers cannot be restricted to solving problems but needs to include the framing of problems. One cannot understand a problem without having a concept of the solution in mind: *"one cannot gather information meaningfully unless one has understood the problem but one cannot understand the problem without information about it"* [Rittel & Webber, 1984]. If one cannot begin one without the other, then the only way to proceed is with both simultaneously.

UNSELFCONSCIOUS CULTURES OF DESIGN

Large-scale design projects are inherently collaborative, ongoing, and evolving. The artifacts produced in these projects must function for years, long after the initial design phase is complete. During this time, the environment in which the artifact functions may change in ways that were not anticipated by the original designers. If the artifact cannot be adapted or evolved by *design-in-use* to its changing environment, it will cease to be useful [Henderson & Kyng, 1991].

Alexander [Alexander, 1964] introduced the distinction between an unselfconscious culture of design and a self-conscious culture of design. In an *unselfconscious* culture of design, the failure or inadequacy of the form leads directly to an action to change or improve it. This closeness of contact between designer and product allows constant rearrangement of unsatisfactory details. In unselfconscious design, breakdown and correction occur side by side; the knowledge to repair breakdowns comes from the knowledge of the user, who is best able to recognize a lack of fit, and how the artifact should be changed to improve its fit to the environment. Table 1 summarizes some of the major distinction between self-conscious and unself-conscious cultures of design.

Putting Owners of Problems in Charge. Ill-defined problems cannot be delegated; therefore the owner(s) of a problem need to be present in incrementally frame the problems, because they have the "authority" to change the problem. If owners of problems are in charge, then b*ackground assumptions do not need to be fully* articulated to avoid to achieve an impossible task [Suchman, 1987]. It is a strength of human experts that they know the larger problem context, which enables them to solve ill-defined design problems, to learn while solving problems, to notice similarities between design problems, and to know when design rules can and should be broken.

Table 1: Comparing Self-conscious and Unself-conscious Cultures of Design

	self-conscious	unself-conscious
definition	an explicit, externalized description of a design exists (theoretical knowledge)	process of slow adaptation and error reduction (situated knowledge)
original association	professionally dominated design, design for others	primitive societies, handmade things, design for self
primary goal	solve problems of others	solve own problems
examples	designed cities: Brasilia, Canberra Microsoft Windows	naturally grown cities: London, Paris Linux
strengths	activities can be delegated; division of labor becomes possible	many small improvements; artifacts well suited to their function; copes with ill-defined problems
weaknesses	many artifacts are ill-suited to the job expected of them	no general theories exist or can be studied (because the activity is not externalized)
requirements	externalized descriptions must exist	owners of problems must be involved because they have relevant, unarticulated knowledge
evaluation criteria	high production value; efficient process; robust; reliable	personally meaningful; pleasant and engaging experience; self-expression
relation with context	context required for the framing of the problem	both problem framing and solving take place within the bigger context

Supporting Unselfconscious Cultures of Design with Domain-Oriented Design Environments. *Domain-oriented design environments (DODEs)* [Fischer, 1994] put owners of problems in charge by supporting human problem-domain interaction rather than just human-computer interaction. The breakdowns users of DODEs will experience include gaps in design knowledge, lack of support for new domain elements, and new rules and guidelines that were not part of the original DODE. These breakdowns cannot be avoided; they are a consequence of the fact that design domains change with time. DODEs support unselfconscious cultures of design with the following tools and mechanisms:

- they support the *co-evolution* of problem framing and problem solving [Nakakoji, 1993];
- they increase the *back-talk* of design situations with critics [Fischer et al., 1998];
- they support *reflection-in-action* by making argumentation serve design [Fischer et al., 1996]; and
- the support the *seeding, evolutionary growth, reseeding process model* to incrementally refine and evolve systems as living entities [Fischer et al., 2001].

Increasing the Back-Talk of Design Artifacts. The core of Schön's framework for reflective practitioner can be summarized as follows: *the designer acts to shape the design situation by creating or modifying design representations, and the situation "talks back" to the designer, revealing unanticipated consequences of the actions. The designer reflects on the actions and consequences by listening to the situation's back-talk, and then plans the next course of action.*

Therefore: design materials and the externalized representations are essential to design as a reflective conversation. Externalized representations uncover *implicit, tacit, and emergent* dimensions of design tasks that designers may not have considered. Externalizing ideas is not a matter of emptying out the mind but of actively reconstructing it, forming new associations, and expressing concepts while lessening the cognitive load required for remembering them [Bruner, 1996].

Critics. While representations can make our thoughts more accessible, it is important to recognize the relationship between the skill and experience of designers and the "back-talk" they receive from the situation. The fact that "*buildings do not speak for themselves*" [Rittel, 1984] reminds us that the meanings and intentions that are "designed into" an artifact are not always self-evident, either to the designer or other observers. *Critiquing systems* [Fischer et al., 1998] monitor the design process and attempt to detect problematic situations. When such a situation is detected, critics notify users and make further information available to help users understand the situation. Critiquing systems allow users to work in a self-directed manner and interrupt only when the users' plans, actions, or products are considered potentially problematic. The role of critics is to inform reflective practitioners, to make them aware of potential problems and help make trade-offs, rather than to design for them.

TRANSCENDING SCHÖN

Schön [Schön, 1992] ends one of his papers with the following challenge: *"The design of design assistants is an approach that has not in the past attracted the best minds in AI. Perhaps the time has come when it can and should do so"*. Schön was interested in developing a *descriptive* account of design activities, illustrating and explaining what designers do, identifying the importance of human collaborations in this process, and arguing for educational changes. He did *not* design and/or build more powerful socio-technical environments that would empower reflective practitioners beyond the possibilities provided by pencil and paper technologies..

But design never *was and never will be independent of the media* used to support the creation of artifacts. What has been true on a very global scale that *"the story of the human race is one of increasing intellectual capability; our brain have gotten no bigger, our bodies no stronger, but there has been incremental creation and evolution of new tools for physical and intellectual work to support more effective ways of distributed work and cognition",* is true for design. Socio-technical environments will empower reflective practitioners to be more effective, to avoid and overcome problems, and learn new things as they go along. Our research has be grounded in Schön's theory in the following way:

- we have *build objects-to-think-with* in the form of demonstration prototypes (e.g. DODEs, critiquing systems);
- we have developed *process innovations* (e.g., meta-design, seeding, evolutionary growth, reseeding process model);
- we have *deployed, used, and evaluated* these prototypes [Bonnardel & Sumner, 1994; Sumner et al., 1997].

With DODEs, we have investigated the following questions:

- How can computational media change the nature of the *reflective conversation* between designer and the materials of the situation [Redmiles, 2002]? Unlike paper, computational media can provide active design materials that allow the situation to talk back to the designer in an explicit manner.
- How can computational media support the *integration of problem framing and problem solving?* By partially externalizing the framing in explicit computational representations such as specification components [Nakakoji, 1993], new ways of supporting design are possible. If the designer's framing of a problem is interpretable by the computer, it allows the computer to detect conflicts between the current design and the framing [Shipman, 1993].
- How can computational media support designers in dealing with *breakdowns*? First, they can help designers to identify breakdowns that they may not be aware of. Second, the occurrence of breakdowns provides an opportunity for learning on demand and reflection-in-action, facilitated by making argumentation serve design [Fischer et al., 1996].

From Reflective Practitioners to Reflective Design Communities. Complex design problems require more knowledge than any single person possesses because the knowledge relevant to a problem is usually distributed among stakeholders. Bringing different and often controversial points of view together to create a shared understanding among these stakeholders can lead to new insights, new ideas, and new artifacts. The challenge for the future will be not only to develop new frameworks, new media, and new social environments to support reflective practitioners but to support *reflective design communities* thereby extending the limitations of the individual human mind. Simon [Simon, 1996] argued that when a domain reaches a point at which the knowledge for skillful professional practice cannot be acquired in a decade, specialization increases, collaboration becomes a necessity, and practitioners make increasing use of media supporting distributed cognition. Design is a prime example of such a domain [Arias et al., 2000].

Issues for further Investigation. More than ten years ago, we articulated the following issues for further investigation [Fischer & Nakakoji, 1992]:

- Are there differences in the performance and quality of the product if the system is used with and without critics?
- What are the tradeoffs between running the system in a critiquing mode or a constraint mode, where the latter prevents certain problems from arising, whereas the former provides designers with opportunities of dealing with breakdowns?
- What are the tradeoffs between different intervention strategies, e.g. the balance between displaying enough information versus the disruption of the work process? When are designers willing to suspend the construction process to access relevant information? Does making information relevant to the task at hand prevent serendipity?
- If an environment can always supply the information that the situation demands, why will users bother to learn the information?
- Under which conditions will designers challenge or extend the knowledge represented in the system? How can they be motivated to do so?
- Should the 'back talk' be embedded directly into the artifact, or handled by a separate discourse? It is conceivable that diving into hypermedia focuses users on other tasks, and takes them out of the situation?
- If information is plentiful, what is scarce? How can information delivery systems be created that make information more relevant to the task at hand?
- To what extent are situations and reflective conversations controlled by media properties?
- How can a balance be achieved between technical rationality (e.g. the use of plans and rules) and reflective action?

It is the author's hope that the CHI workshop will provide *many new answers from all the participants to the issues.*

ACKNOWLEDGEMENTS

The ideas, framework, and systems briefly presented in this contribution have been developed *collaboratively by the reflective design community* of past and current members of the Center for Lifelong Learning and Design.

REFERENCES

Alexander, C. (1964) The Synthesis of Form, Harvard University Press, Cambridge, MA.

Arias, E. G., Eden, H., Fischer, G., Gorman, A., & Scharff, E. (2000) "Transcending the Individual Human Mind—Creating Shared Understanding through Collaborative Design," ACM Transactions on Computer Human-Interaction, 7(1), pp. 84-113.

Bonnardel, N., & Sumner, T. (1994) "From System Development to System Assessment: Exploratory Study of the Activity of Professional Designers." In Proceedings of the 7th European Conference on Cognitive Ergonomics (Bonn, Germany), pp. 23-36.

Bruner, J. (1996) The Culture of Education, Harvard University Press, Cambridge, MA.

Fischer, G. (1994) "Domain-Oriented Design Environments," Automated Software Engineering, 1(2), pp. 177-203.

Fischer, G., Grudin, J., McCall, R., Ostwald, J., Redmiles, D., Reeves, B., & Shipman, F. (2001) "Seeding, Evolutionary Growth and Reseeding: The Incremental Development of Collaborative Design Environments." In G. M. Olson, T. W. Malone, & J. B. Smith (Eds.), Coordination Theory and Collaboration Technology, Lawrence Erlbaum Associates, Mahwah, NJ, pp. 447-472.

Fischer, G., Lemke, A. C., McCall, R., & Morch, A. (1996) "Making Argumentation Serve Design." In T. Moran, & J. Carrol (Eds.), Design Rationale: Concepts, Techniques, and Use, Lawrence Erlbaum and Associates, Mahwah, NJ, pp. 267-293.

Fischer, G., & Nakakoji, K. (1992) "Beyond the Macho Approach of Artificial Intelligence: Empower Human Designers - Do Not Replace Them," Knowledge-Based Systems Journal, Special Issue on AI in Design, 5(1), pp. 15-30.

Fischer, G., Nakakoji, K., Ostwald, J., Stahl, G., & Sumner, T. (1998) "Embedding Critics in Design Environments." In M. T. Maybury, & W. Wahlster (Eds.), Readings in Intelligent User Interfaces, Morgan Kaufmann, San Francisco, pp. 537-559.

Henderson, A., & Kyng, M. (1991) "There's No Place Like Home: Continuing Design in Use." In J. Greenbaum, & M. Kyng (Eds.), Design at Work: Cooperative Design of Computer Systems, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, pp. 219-240.

L3D (2004) Center for LifeLong Learning and Design , University of Colorado, Boulder, Available at http://www.cs.colorado.edu/~l3d/.

Nakakoji, K. (1993) Increasing Shared Understanding of a Design Task Between Designers and Design Environments: The Role of a Specification Component, Ph.D. Dissertation, University of Colorado at Boulder.

Redmiles, D. (2002) "Supporting the End Users' Views," Proceeding of Advanced Visual Interfaces Conference (AVI 2002, May), Trento, Italy, pp. 34-42.

Rittel, H. (1984) "Second-Generation Design Methods." In N. Cross (Ed.), Developments in Design Methodology, John Wiley & Sons, New York, pp. 317-327.

Rittel, H., & Webber, M. M. (1984) "Planning Problems are Wicked Problems." In N. Cross (Ed.), Developments in Design Methodology, John Wiley & Sons, New York, pp. 135-144.

Schön, D. (1992) "Designing as reflective conversation with the materials of a design situation," Knowledge-Based Systems Journal, Special Issue on AI in Design, 5(1), pp. 3-14.

Schön, D. A. (1983) The Reflective Practitioner: How Professionals Think in Action, Basic Books, New York.

Shipman, F. (1993) Supporting Knowledge-Base Evolution with Incremental Formalization, Ph.D. Dissertation, University of Colorado at Boulder.

Simon, H. A. (1996) The Sciences of the Artificial, third ed., The MIT Press, Cambridge, MA.

Suchman, L. A. (1987) Plans and Situated Actions, Cambridge University Press, Cambridge, UK.

Sumner, T., Bonnardel, N., & Kallak, B. H. (1997) "The Cognitive Ergonomics of Knowledge-Based Design Support Systems." In S. Pemberton (Ed.), Proceedings of CHI 97 Conference on Human Factors in Computing Systems, ACM/Addison-Wesley, pp. 83-90.

The use of cognitive causal mapping as an aid to professional reflection

Robert T.Hughes, Abdullah Al Shebab, Marian Eastwood
School of Computing, Mathematical and Information Sciences,
University of Brighton, Brighton, BN2 4GJ United Kingdom

Email r.t.hughes@bton.ac.uk

Our background is broadly in software engineering, typically but not exclusively within information systems domains. Our individual research interests have included software effort estimation, software project risk, and development process models which may reduce risk. Two of us are committed to teaching students at undergraduate and graduate levels. As busy academics, we like research challenges that have the potential to enrich our students' learning experiences, as well as, in the longer term, being of interest in the wider world.

Schön's work resonates with those teaching vocational subjects such as software engineering, but is also frustrating. The observation of professionals at work can offer some insights into practice, yet much remains concealed or unclear. As it says in one case study: *'…the Resident discerns in the Supervisor's performance a knowing-in-practice that he values, but is frustrated in his attempt to grasp it*'. Schon (1983). One of us (Hughes 1996) felt a similar frustration during an attempt to elicit how software developers produced estimates of development effort.

As teachers we would like in some way to capture practitioners' expertise so that it can be presented as 'professional knowledge' to our students. One recent emphasis in the UK has been on 'life-long learning, a term often difficult to interpret in practice, but generally held to be a 'good thing'. Our interpretation is that one aspect of this is that students ought to be taught to learn from their experience, as well as from pre-packages modules of learning material, and that this will require them to be able to reflect on their practical experiences.

To this end, as educators, we have recently focussed on the production of software prototypes by students as a vehicle for learning (Eastwood and Hughes, 2004). In industry, prototypes are long-established as a way of learning and reducing uncertainty in the context of a specific project (see, for example, Lichter et al. 1997). This type of 'industrial' learning needs to be focussed and fast. However, at the same time, practitioners are acquiring skills and knowledge that can be applied to later projects. This longer term learning is unlikely to be a concern of the project manager who has more immediate anxieties. One can therefore idealise a number of learning cycles of different lengths progressing concurrently.

Higher education ought to have more concern with the longer cycle learning. The use of prototyping in this context should allow the student to exercise some short cycle skills in software development and also, one hopes, learn some more abstract, long cycle, lessons about design and project management.

The use of a prototype to solve a particular local problem is a process that is very visible, not just to tutors in an academic environment who may wish to assess it, but to users in the 'real world'. A more difficult problem is to assess long cycle learning. A 'traditional' way has been to compel the student to complete a reflective report describing the way that the particular problem was solved. Our experience is, however, that the 'better' students will often respond by describing the process which they believe the tutors want to hear, normally about the slavish following of some standard methodology, rather than genuinely reflecting on their actual experience.

If one turns to the 'real world' of software development a striking feature is the prevalence of failed projects (see, for example Flowers, 1996). Part of the reason for this may be the innate difficulties of software, as identified by Brooks (1995), namely its relative invisibility, its complexity, the flexibility that makes it have to conform to the whims of clients, however illogical, and also make it subject to constant change. Despite this, one can still sometimes think that if, in this field, the 'professionals' are building up Schonian 'repertoires' based on past experience, then the process does not seem to be overwhelming successful. We suspect that one reasons for this prevalence of failure is that all types of large organisation are likely at some point to be subject to information systems implementation projects, and we cannot expect all managers in all organizations to turn themselves into project management professionals at a moment's notice. We also suspect that there is often a lack of longer cycle organizational learning. In such environments we suspect that project failure is often caused by influential stakeholders having preconceptions about the factors at work that will influence the success or failure of the project, and then these preconceptions not being matched by reality.

Our proposal for ameliorating this mismatch between the preconceived and the actual is to attempt to make more explicit to stakeholders the nature of these preconceptions by employing collaborative cognitive causal mapping techniques. These are influenced by Kelly's constructivist model of human motivation (Kelly, 1955). This sees the being in the world trying to predict and control the outcome of events by using past experience to identify those variables that are likely to dictate the future course of events.

These variables are perceived as dichotomous indicators that reflect the degree to which a concept has a value between two extremes, for example, large and small, cheap and expensive, fast and slow. These variables are represented as nodes in a network, and directed arcs between the nodes indicate that one variable can influence the value of another. For example, Figure 1 reflects the thinking that incomplete requirements may lead to the functionality of an application being increased to cover the missing requirement when found. This in turn increases the overall cost

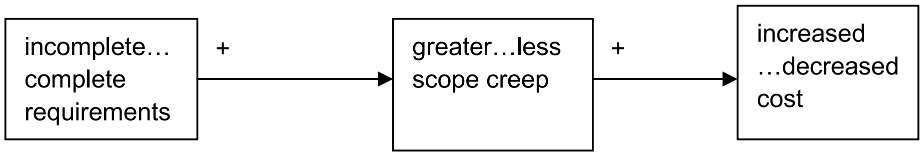


Figure 1 A simplistic causal model

What might happen is that in order to overcome this unwelcome chain of events, a prototype is planned, which it is hoped will help elicit a full set of requirements from the users - see Figure 2.

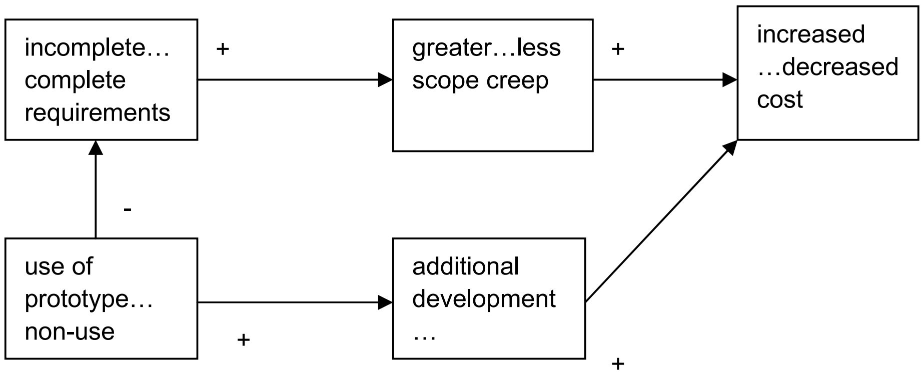

incomplete… complete requirements
+
greater…less scope creep
+
increased …decreased cost
-
use of prototype… non-use
+
additional development …
+

Figure 2 Adding the use of a prototype

The use of the prototype, while reducing the possibility of incomplete requirements, will also contribute to cost as it requires additional effort to create the prototype. Where there is a negative, inhibiting, link in a causal chain then, unless there is a compensating second negative link, the causal chain between two nodes will be a negative. Thus the causal chain between ‘use of the prototype’ via ‘incomplete requirements’ to ‘increased…decreased cost’ is a negative one, that is, the use of prototypes should decrease costs. However, the second causal chain between ‘user of prototype’ and ‘increased…decreased cost’ is a positive one: the prototype will add to costs. This is an example of an *unbalanced* causal map. Unbalanced causal maps indicate that some outcomes are uncertain, which indicates that there is an element of risk. Another diagnostic feature of causal maps is that they can detect feedback loops and also *unstable* systems where an increase in an external variable can cause some of the variables to grow in an uncontrolled manner. For example, a stream of changes, where there is no effective change control mechanism, might cause runaway development costs.

The production of causal maps can be done retrospectively, at the post-mortem stage of a development project, in order to diagnose the reasons for unsatisfactory and satisfactory outcomes of the project. It can also be used at the beginning of a project to help stakeholders identify potential difficulties and possible policies to reduce those difficulties. Clearly using the output from post mortems from previous projects to inform the risk analysis for future projects could facilitate learning, at both the level of the individual professional and at the organizational level.

Cognitive mapping approaches are well established - one early use (Axelrod, 1976) was to analyse how diplomats and government officials decided and applied policies, particularly in the field of foreign policy. In the field of project management and operational research, they have been used extensively - see, for example, Eden 1988, Eden et al. 1992, Brown 1992, Williams et al 1995.

Often cognitive maps are used as a preliminary stage to the creation of a quantitative systems dynamics model (see, for example, Abdel-Hamid and Madnick 1986, Williams et al 1995). The addition of quantitative data could, for example, resolve whether the use of a prototype in the fragment in Figure 2 would be worthwhile. Quantitative considerations can enrich the causal models by reflecting the strength of the influence of one variable upon another; they can also indicate the presence of time delays in the influence of one variable being felt by another. Senge (1990) has advocated the use of animated, quantitative models as a means of fostering systems thinking in organizations. The ability of such models to demonstrate the long term effects of policies is particularly stressed by such advocates. We are, however, rather cautious about attempting to enhance the modelling to take on quantitative aspects, despite our computer science backgrounds. Partly this is because we are aware of the pitfalls associated with measurement (Hughes 2000). It is also because we have a suspicion that showing that a computer model produces the results it was programmed to produce might demonstrate skilful programming rather than any underlying new knowledge.

We are currently using these techniques in research into project risk, by carrying out retrospective mapping of industrial projects, as an aid to the diagnosis of project short-comings. We are also planning to train our undergraduate and graduate students in the technique so that the can apply the approach both at the planning and reflective stages of their student projects as an aid to learning. Masters students in information systems and software engineering have, for example, to conduct a project for an outside client as the last element of their courses. We would like to require these students to produce cognitive maps as a way of justifying their planned approach to the project and also at the end of the project to support their reflection on the project.

References

Abdel-Hamid, T.K. and S.E.Madnick. 1986 Impact of schedule estimation on software project behavior. *IEEE Software* 34(4) 70-5

Axelrod, R. (ed.) 1976 *Structure of decision: The cognitive maps of political elites*. Princeton University Press, Princeton, NJ

Brooks, F.P. 1987. No silver bullet, essence and accidents of software engineering. *IEEE Computer* April 10-19

Brown, S. 1992. Cognitive mapping and repertory grids for qualitative survey research: some comparative observations. *Journal of Management Studies* 29(3) 287-307

Eastwood, M. and Hughes R.T. 2004. Software prototyping as experiential learning. SQM/Inspire conference. Canterbury, Kent. April

Eden, C. 1988 Cognitive *mapping European Journal of Operational Research* 36 1-13

Eden, C., F.Ackermann, and S.Cropper. 1992. The analysis of causal maps. Journal of Management Studies 29(3) 309-324

Flowers, S., 1996 *Software failure, management* failure. Wiley and Sons, Chichester

Hughes, R.T. 1996 Expert judgement as an estimating method *Information and Software Technology* 38(3) 67-75

Hughes, R.T. 2000. *Practical software measurement*. McGraw-Hill, Maidenhead

Kelly, G.A. 1955 *The psychology of human constructs. Volume 1.* W.W.Norton and Co. New York

Lichter et al 1997. Prototyping in industrial software projects - bridging the gap between theory and practice. *Proceedings of the 15th International Conference on Software Engineering* May 1997 221-229

Schon. D.A. 1983. *The reflective practitioner: how practitioners think in action*. Basic Books

Senge, P.M. 1990. *The fifth discipline*. Random House, London

Williams, T., C. Eden, F. Ackermann. and A.Tait. 1995 The effect of design changes and delays on project costs. *Journal of the Operational Research Society*. 46 809-18

Designing for Reflective Practitioners

Kumiyo Nakakoji
University of Tokyo
4-6-1 Komaba,
Meguro, Tokyo,
153-8904 Japan
kumiyo@kid.rcast.u-tokyo.ac.jp

Session: New Methods and Techniques for Software Development

Meta Architecture for Intelligent Information Systems

- Possition paper -

Milorad Tošić
University of Niš, Faculty of Electronics Engineering
18000 Niš, Serbia & Montenegro
e-mail: mbtosic@elfak.ni.ac.yu

ABSTRACT
In this paper, we identify meta architecture of the Intelligent Information Systems as an open research problem addressing some of the big challenges that the Information Technology industry is now facing in the environment of current trends on the turbulent global market. As the foundation of future research, the proposed metamodel is to be evaluated against different application areas as well as related information technologies, starting with the area of social computing.

Author Keywords
Meta architecture, social computing.

ACM Classification Keywords
H.5.3 Group and Organization Interfaces

INTRODUCTION

Advancements in the Information Technology (IT) are rapidly becoming leading force in human society development. As a consequence, the mutual impact is more and more evident where IT not only changes the way humans live and work (including businesses, social life, government, entertainment, etc.) but it also suffers tremendous pressure to deliver human-oriented value that is actually needed. We have witnessed last years how big the expectations for IT could be: ".com" bubble spectacularly raised immediately after emergence of first signs of the great value that Internet, as the global IT infrastructure, brings along.

Intelligent Information Systems (IIS) represent the next generation of information systems embodying knowledge that allows them to exhibit intelligent behavior, cooperate with users and other systems in problem solving, discovery, access, retrieval and manipulation of a wide variety of multimedia data and knowledge, and reason under uncertainty 6. The IIS is no more only passive (collecting information, processing and presenting it in a structured way as a classical information system does) but also open, global, interactive and reflective (it is an integral part of a global environment, it reasons about behavior, communicates and collaborates, has the purpose and mission, etc.). This new setting presents fundamentally new challenges to the IT research community that is no more closed and "elitistic" but expected to provide pervasive open platforms for heterogeneous and multi-disciplinary work. Having a historical perspective, we can see IT advancing in a predictable sequence of five "mega waves", where we currently are at the end of the third and at the beginning of the emerging fourth mega wave 5.

In this paper we present the vision of what the value-add frontier of IT development will be on the forth "mega wave". We identify IIS as the central point in the world of content, systems, organizations, communities, and businesses converging into a global virtual infospace. We also argue that the meta architecture of the IIS is the most promising approach to provide necessary "virtualness", separation of concerns, and value-add in the global infospace environment. Hence, we propose metamodel of *the Intelligent Information Systems Virtual Machine (IISVM)* - a universal platform for transparent from-value-to-machine design.

ASPECT ORIENTED IIS DESIGN

For design of the IIS systems we adopt best practices form the Aspect Oriented Programming (AOP) 4 discipline, with a goal to generalize separation of concerns principle up to the strategy level. The principle empower us to cope with high problem complexity by adopting approaches already proven in the software systems design, (such as component-based approach 9, meta-programming 1, etc.) and applying them transparently across whole chain of concerns up to the highest abstraction levels (such as value creation 12, resource planning 11 and coordination 14, and e-Business models 13). As a result of the approach, we envision *the Intelligent Information Systems Virtual Machine (IISVM)*: A universal, transparent, and pervasive IT platform for development of reflective value-driven applications.

Technology aspect of the IIS is increasingly correlated to the areas such as 6: discovery of knowledge from large data collections; providing cooperative support to users in complex query formulation and refinement; access,

level M3:	Meta meta classes	meta-metamodel (MOF model)
level M2:	Meta classes	metamodel
level M1:	Classes	model
level M0:	Objects	information

Figure 1: MOF four layer metadata architecture

retrieval, storage and management of large collections of (multimedia) data and knowledge; information integration from multiple heterogeneous data and knowledge sources; behavior and information unity in virtual systems, and reasoning about information under uncertain conditions. Having in mind ultimate impact of the global network, the emerging need for new tools and techniques for management of these dynamic and evolving information spaces existing on a global scale over the Internet is evident.

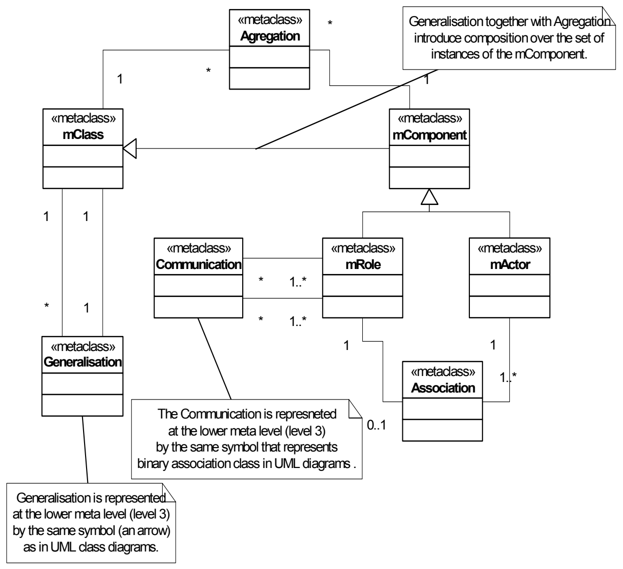
Value (business and social) aspect of the IIS has evolved, due to the global acceptance of the Internet, from very limited impact (when computing centers were used for IT support of big enterprises needs only) to increasingly high (e-government, e-communities, e-business, e-learning, etc.). Consequently, IIS is required not only to automate information processing, storage and distribution but also to reason about issues such as knowledge sharing 9,13, value creation 12, and social impact 15.

Design aspect of the IIS includes interoperability, platform independence, reusability, concurrency and abstraction. This aspect has been in the focus of research in IT community for a long time 8, resulting in emerging technologies such as Model Driven Architecture 16 and Service Oriented Architecture 20. Our belief is that the design aspect will benefit the most from the proposed approach by adopting mechanisms from other aspects as design components (for example, applying auctions and value-based formal business models to the collaboration of software components and platforms, e.g. see 18) while ”borrowing” well-known proven design practices to other aspects as well.

META ARCHITECTURE

We base meta architecture of the IISVM on the Meta Object Facility (MOF) four layer metadata architecture (Figure 1:) 3. We adopt the level M3 from the MOF model and the reflection on the same level 1,3, while building the proposed meta architecture on the M2 level.

The proposed metamodel is shown in Figure 2: . We may distinguish “object” and “relation” meta meta objects (instances of meta meta classes): mClass, mComponent, mActor, and mRole may be interpreted as “object” meta classes, while Generalization, Agregation, Communication and Association as relation meta classes.



«metaclass» Agregation
Generalisation together with Agregation introduce composition over the set of instances of the mComponent.
«metaclass» mClass
«metaclass» mComponent
«metaclass» Communication
«metaclass» mRole
«metaclass» mActor
«metaclass» Generalisation
«metaclass» Association
The Communication is represneted at the lower meta level (level 3) by the same symbol that represents binary association class in UML diagrams .
Generalisation is represented at the lower meta level (level 3) by the same symbol (an arrow) as in UML class diagrams.
1
*
1..*
0..1

Figure 2: The IIS metamodel

The metamodel facilitates unified approach to different aspects of the IIS design (as previously described): mClass, Aggregation and Generalization support the technology aspect following object-oriented programming approach; mComponent, Communication, and Association support design aspect, while mActor, and mRole support business and social aspect. The fact that all that meta classes belong to the same metamodel enable us to reason about the system in a transparent and unified way across all aspects of the IIS.

Let us given the set of components and the set of classes of objects from the set of components. Following well-established theory of the object-oriented programming, represented with the UML standardization initiative and the MOF architecture 3, there is an inheritance relation (the generalization meta class in Figure 2:) introduced in the set of classes. For example, in a system where User, Mediator, ServiceProvider are possible components, all of IndividualUser, EnterpriseUser, and Administrator classes may share the common base class User.

It should be noted that the mComponent is specialized into two sub-classes: mRole and mActor. The idea is to separate architectural concerns of behavior from the interaction concerns: The only component that may have associated actions and/or behavior is the mActor class, while mRole component is the only one that may be involved in an interaction by means of the associated Communication. In this way, an actor that have specified behavior and performs some actions may interact with the rest of the system only if it is encountered in an interaction by accepting some role in the system. An actor may have multiple roles and multiple actors may “play” the same role.

In the proposed architecture, the Communication is not a component (it doesn't inherit mComponent class). In this way, we provide the crucially important flexibility for dealing with reflectiveness and interaction. For example, let us consider a simple client-server setting. Traditionally, we have two roles (client and server) assigned to two components (e.g. web browser and web server, respectively) with implicitly assumed communication capabilities. However, the proposed architecture enables us to identify the third role: the client-server interaction role. The client and server roles do not communicate directly, but by means of the interaction role. In this way, we are able to implement the interaction role by different components that may act as a network communication protocol 20, an auction based negotiation 18,2, or a very complex social interaction 15,23.

SOCIAL COMPUTING: AN APPLICATION CASE STUDY

Social computing is an emerging inter-disciplinary research area addressing synergy potential of social aspects in the information society. It builds on the mass adoption of information technologies: Current estimates of Internet users hover around 200 million, with one billion users anticipated by 2005, together with the 'Net's technology-mediated communication services that provide unique opportunities for extending many human pursuits 21. Following the trend towards fully pervasive computing where end-points of information streams may be humans as well as machines, it is evident that the social dimension becomes crucially important 22,24. Also, we are in a transition from traditional understanding of computers as "things that think" (what computers can do?) to much more powerful "things that make us smart" (what people and computers can do together?) 23.

We believe that the proposed IISVM architecture may answer the challenge of achieving full power of the social aspect of the computing in general. Particularly, it is possible to develop solutions for efficient addressing of social reflection of the computing in diverse application fields, such as e.g. education 24, enterprise applications, etc. Our current research is focused on development of the concept of "recommendation" in the social computing environment, as an application testbed for the proposed architecture.

REFERENCES

1. Ira R. Forman, Scott H. Danforth, *"Putting metaclasses to work: a new dimension in object-oriented programming"*, Addison Wesley Longman, Inc., 1999.
2. B. Wydrowski and M. Zukerman, "QoS in Best Effort Networks", *IEEE Communication Magazine*, December 2002, pp.44-49.
3. Object Management Group, *Meta Object Facility (MOF)*, 1.4, http://www.omg.org/, (accessed Sept. 2003)
4. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. "Aspect-oriented programming". In *ECOOP'97---Object-Oriented Programming, 11th European Conference*, LNCS 1241, pages 220--242, 1997. http://citeseer.nj.nec.com/kiczales97aspectoriented.html
5. Kampas, P.J., "Road Map to the E-Revolution", *Information Systems Management*, vol.17, no.2, Spring 2000. Reprinted in IEEE Engineering Management Review, First Quarter 2001, pp.81-93.
6. *Journal of Intelligent Information Systems*: Integrating Artificial Intelligence and Database technologies, Mission and Scope, http://www.isse.gmu.edu/JIIS/JIIS_Folder/Mission.Scope.html (accessed Sept. 2003.)
7. M. McKinlay, Z. Tari, "DynWES – A Dynamic and Iteroperable protocol for Web Services", *Third International Symposium on Electronic Commerce (ISEC'02)*, October 18-19 2002, Research Triangle Park, North Carolina, pp.74-83
8. Borghoff, U.M., Schlichter, J.H., *"Computer-Supported Cooperative Work: Introduction to Distributed Applications"*, Springer, 2000.
9. McIlraith, S.A., Son, T.C., Zeng, H., "Semantic Web Services", *IEEE Intelligent Systems*, March/April 2001, pp.46-53.
10. Szyperski, C., "*Component Software: Beyond Object-Oriented Programming*", Addison-Wesley, Harlow, UK, 1998
11. Ghallab, M., et al., "PDDL: The Planning Domain Definition Language, Version 1.2", Tech. Report CVC TR 98 003/DCS TR 1165, Yale Center for Computational Vision and Control, Yale Univ., New Haven, Conn., 1998.
12. Gordijn, J., "e-Business Model Ontologies", contribution to *"Value creation from e-business models"*, Wendy Curry, http://www.math.vu.nl/~gordijn/research.htm, to appear (accessed Sept. 2003)
13. Osterwalder, A., Pigneur, Y., "Towards Strategy and Information Systems Alignment through a Business Model Ontology", to appear in the *Proceedings of the 23rd Annual International Conference of the Strategic Management Society (SMS)* 2003, Baltimore
14. Fan. M., Stallaert, J., Whinston, A.B., "Decentralized Mechanism Design for Supply Chain Organizations Using an Auction Market", *Information Systems Research*, Vol. 14, No. 1, March 2003, pp.1-22.
15. Miranda, S.M., Saunders, C.S., "The Social Construction of Meaning: An Alternative Perspective On Information Sharing", *Information Systems Research*, Vol. 14, No. 1, March 2003.

16. *OMG Model Driven Architecture*, http://www.omg.org/mda/

17. Fan, M., Stallaert, J., Whinston, A.B., "The Adoption and Design Methodologies of Component-Based Enterprise Systems," *European Journal of Information Systems*, 9 (1), 2000, 25-35.

18. Wellman, M.P., Market-Oriented Programming: Some Early Lessons, In S. Clearwater (ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

19. Lars Marius Garshol, Living with topic maps and RDF, Ontopia, (accessed Dec. 2003) http://www.ontopia.net/topicmaps/materials/tmrdf.html

20. W3C, "Web Services Architecture", *W3C Working Draft* 8 August 2003, http://www.w3.org/TR/2003/WD-ws-arch-20030808/

21. Patsy Clarke, If the Internet is the solution – what is the problem?, http:// www.und.ac.za/users/clarke/phd/p@c1.pdf

22. Erickson, T., Halverson, C., Kellogg, W.A., Laff, M., Malkin, P., and Wolf, T. Loops: Designing a Web-Based Environment for Persistent, Semi-Structured Conversation (draft paper) http://www.research.ibm.com/SocialComputing/Papers/Loops_draft_Nov02.pdf (accessed 24. Dec. 2003)

23. Gerhard Fischer, "Communities of Interest: Learning through the Interaction of Multiple Knowledge Systems", *Proceedings of the 24th IRIS Conference* (eds: S. Bjornestad, R. Moe, A. Morch, A. Opdahl), August 2001, Ulvik, Department of Information Science, Bergen, Norway, pp 1-14

24. Donald Schön, "Educating the Reflective Practitioner", meeting of the American Educational Research Association, 1987, Washington, DC (accessed 23. Dec. 2003. http://educ.queensu.ca/~russellt/howteach/schon87.htm)

25. Ackerman, M., and Starr, B., "Social activity indicators: interface components for CSCW systems," in *Proceedings of The 8th ACM syrnposiurn on User interface and software technology*, (Pittsburgh, PA, USA), ACM Press, 1995.

Reflective practitioners in software design. Case study :Extreme programming

Cestmir Halbich, Department of information technologies, CUA Prague

Abstract: The position paper comes out from Donald Schoen's ideas about reflective practitioners and describes author's experience in area of computer science. The software gap is mentioned and methods for its overbridging too. At the case study is illustrated Schoen's approach to designer's activity and benefit for improving effectiveness and efficiency of the software design process by reflective practitioner's approach. In the conclusion the advantages and disadvantages are described.

Key words: reflective practitioner, extreme programming, software gap

By Schoen designers use design representations to create a virtual "design world" [Schoen 1992] in which the objects and relationships of the design situation are named and made explicit. The situation that the designer constructs with representations enables new understandings of the design problem by allowing the designer to see ideas that before existed only tacitly, and to understand implications of the design situation that did not exist before constructing the representations. In accordance with some authors we can say that putting ideas down on paper is not a matter of emptying out the mind but of actively reconstructing it, then forming new associations, and expressing concepts in pictorial, linguistic, or any explicit representational forms. Designers work mainly with matter, software developers work usually with algorithms, but the analogy is evident.

Knowledge is constructed in programming through an interaction between the programmer's understanding of the design situation and the representations the programmer's creates. Design theorist Donald Schoen characterises the relationship between the designer and design representations as a "reflective conversation with the materials of the situation" [Schoen 1983], in which the designer acts to shape the design situation by creating or modifying design representations, and the situation talks back to the designer, revealing unanticipated consequences of the design actions. By analogy the programmer reflects on the actions and consequences to understand the situation's back-talk, and then plans the next course of action. The software design process is driven by the interaction between the programmer and design algorithms, rather than by following a pre-planned solution approach (see Figure 1).

Knowledge is constructed in software design through repeated cycles of representing and interpreting the design situation. Programmers construct the design situation that talks back in the form of a breakdown. This back talk is interpreted, or reflected upon, to activate

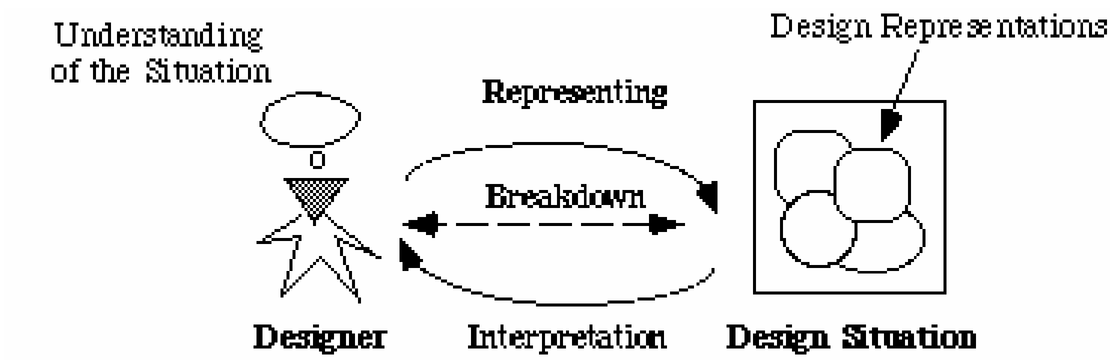


Figure 1. Knowledge Construction in software Design

new understandings of the design situation. In this situation is very useful to use certain formal methodology to improve the effectiveness and efficiency of the software design process.

Already at the conference NATO 1968 the existence of the gap- “gap, between what was hoped for from a complex software systems, and what was typically achieved.” There is a widening gap between ambitions and achievements in software engineering. “This gap appears in several dimensions: between promises to users and performance achieved by software, between what seems to be ultimately possible and what is achievable now and between estimates of software costs and expenditures. The gap is arising at a time when the consequences of software failure in all its aspects are becoming increasingly serious.“ [NATO 1968, p.70]. Since this time amount of different methodologies was invented. One of these is so called extreme programming [Beck 1999].

By author’s opinion the extreme programming is very closely joined with Donald Schoen’s ideas about reflective practitioners. Contrary to some other programmer’s techniques extreme programming closely collaborates with users in the all phases of software design. Some features of extreme programming are discussed below.

Extreme Programming (XP) is actually a deliberate and disciplined approach to software development. It has already been proven at many companies of all different sizes and industries world wide. XP is successful because it stresses customer satisfaction. The methodology is designed to deliver the software your customer needs when it is needed. XP empowers your developers to confidently respond to changing customer requirements, even late in the life cycle. This methodology also emphasises team work. Managers, customers, and developers are all part of a team dedicated to delivering quality software. XP implements a simple, yet effective way to enable groupware style development. XP improves a software project in four essential ways; communication, simplicity, feedback, and courage. XP programmers communicate with their customers and fellow programmers. They keep their design simple and clean. They get feedback by testing their software starting on day one. They deliver the system to the customers as early as possible and implement changes as suggested.

Now we in short describe main features of all phases of software design by extreme programming.

Features in Planning

User stories are written. Release planning creates the schedule. Make frequent small releases. The project velocity is measured. The project is divided into iterations. Iteration planning starts each iteration. Move people around. A stand-up meeting starts each day. Fix XP when it breaks.

Features in Designing

Simplicity. Choose a system metaphor. Use CRC-cards for design sessions. Create spike solutions to reduce risk. No functionality is added early. Refactor whenever and wherever possible.

Features in Coding

The customer is always available. Code must be written to agreed standards. Code the unit test first. All production code is pair programmed. Only one pair integrates code at a time. Integrate often. Use collective code ownership. Leave optimisation till last. No overtime.

Features of Testing

All code must have unit tests. All code must pass all unit tests before it can be released. When a bug is found tests are created. Acceptance tests are run often and the score is published.

Now we describe some details about pair programming. All code to be included in a production release is created by two people working together at a single computer. Pair programming increases software quality without impacting time to deliver. It is counter intuitive, but two people working at a single computer will add as much functionality as two working separately except that it will be much higher in quality. With increased quality comes big savings later in the project. The best way to pair program is to just sit side by side in front of the monitor. Slide the key board and mouse back and forth. One person types and thinks tactically about the method being created, while the other thinks strategically about how that method fits into the class. It takes time to get used to pair programming so don't worry if it feels awkward at first.

Conclusion

We have good personal experience with implementing of the extreme programming in the practice. Reflective practitioner method improves added value of the collaboration in software design process by iterations and feedback loops with customers and end users (see Figure 2). The case study's Czech company has good experience with software design of small, medium and large information systems and work with small and medium work teams.

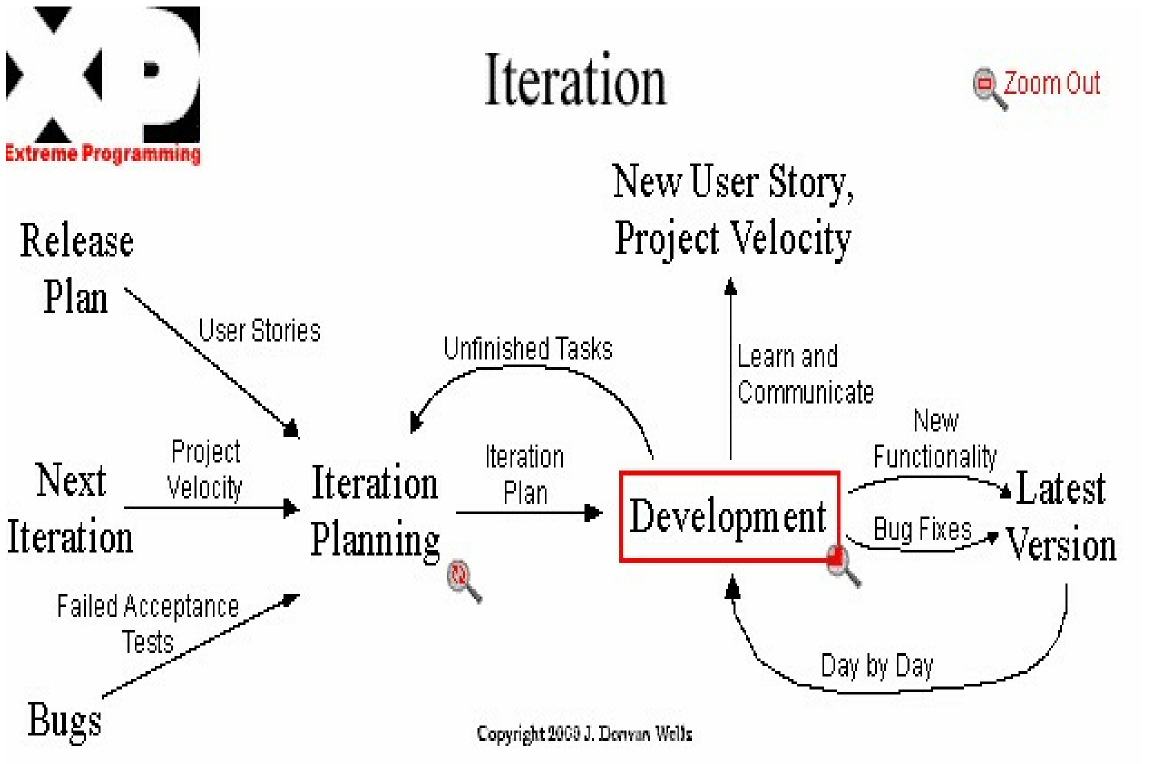


Figure 2. Iterations and feedback lops in extreme programming

References

NATO SOFTWARE ENGINEERING CONFERENCE 1968, Garmisch, 1968

SCHOEN, D.A. The Reflective Practitioner: How Professionals Think in Action, Basic Books, New York, 1983, ISBN: 0465068782

SCHOEN, D. "Designing as reflective conversation with the materials of a design situation," Knowledge-Based Systems Journal - Special Issue on AI in Design, Vol. 5, No. 1, 1992, pp. 3-14

BECK, Kent: Extreme Programming Explained: Embrace Change, 224 pages ; Addison-Wesley Pub Co; 1st edition,1999, ISBN: 0201616416

The Reflective Practitioner Perspective in Software Engineering

Position Paper
CHI 2004 One Day Workshop - Designing for Reflective Practitioners

Orit Hazzan[1] and Jim Tomayko[2]

[1]Department of Education in Technology and Science, Technion - IIT, Haifa 32000, Israel
oritha@tx.technion.ac.il

[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, U.S.A.
jet@cs.cmu.edu

This position paper is based on our articles:

Hazzan, O. (2002). The reflective practitioner perspective in software engineering education, *The Journal of Systems and Software* **63**(3), pp. 161-171.
Hazzan, O. and Tomayko, J. (2003). The reflective practitioner perspective in eXtreme Programming, *Proceedings of the XP Agile Universe 2003*, New Orleans, Louisiana, USA, pp. 51-61.

1. Introduction

This position paper focuses on the application of the Reflective Practitioner (RP) perspective to the profession of Software Engineering (SE). The RP perspective, introduced by Donald Schön (1983, 1987), guides professional people (architects, managers, musicians and others) to rethink and examine their professional creations during and after the accomplishment of the creation process. The working assumption is that such a reflection improves the proficiency and performance within such professions. Analysis of the field of SE and of the kind of work that software engineers usually accomplish, supports the adoption of the RP perspective to SE.

Specifically, this position paper focuses on the construction of ladders of reflection that may serve as a means that supports one's thinking in terms of different levels of abstraction. Indeed, one message that is conveyed in this position paper is that the transition between levels of abstraction is an important skill for software developers. It is proposed that developers' experience in the construction of ladders of reflection may improve their performance in the process of software development.

2. Software Engineering as Reflective Practice

The importance of reflection as a habit-of-mind in the context of SE is derived mainly from two factors: The first factor is the complexity involved in developing software systems, regardless of whether one examines this complexity from an engineering, social or cognitive point of view; the second factor is the crucial role of communication among teammates for the success of developing a software system. The first factor suggests that one must improve one’s understanding about one’s own mental processes. One way to do this is by adopting a reflective mode of thinking. The second factor implies that in order to improve the communication within software development teams, one must increase one’s awareness of one’s own mental processes as well as of the mental processes of others.

3. Ladders of Reflection

This position paper suggests adopting the reflective practice perspective as a cognitive tool which may help software engineers in developing software systems. For this aim we propose to use what Schön (1987) describes by the term ladder of reflection:

> *We can [...] introduce another dimension of analysis* [for the chain of reciprocal actions and reflections that make up the dialogue of student and coach]. *We can begin with a straightforward map of interventions and responses, a vertical dimension according to which higher levels of activity are "meta" to those below. To move "up", in this sense, is to move from an activity to reflection on that activity; to move "down" is to move from reflection to an action that enacts reflection. The levels of action and reflection on action can be seen as the rungs of a ladder. Climbing up the ladder, one makes what has happened at the rung below an object of reflection.* (Schön, 1987, p.114)

The ladder of reflection described in Schön's quote refers to the architecture studio in which a coach guides his or her students. In what follows we attempt to explore how such a ladder of reflection may be associated with software engineering processes. Section 4 presents two ladders of reflection.

4. The Reflective Practice Perspective in Extreme Programming

It seems that an RP approach fits very well to Extreme Programming (XP) (Beck, 2000), since XP emphasizes learning through reflection processes. For example, the estimation of the team's velocity is improved from project to project based on a reflective process; when a pair is engaged in a pair programming session, the navigator reflects on the drivers' coding. Thus, it seems that one of the implicit XP guidelines is reflection. Still, as far as we know, reflection is not outlined in the XP practices themselves.

RP is not explicitly directed to code production but in the long term it may improve code production and quality. As XP incorporates activities that are not directly oriented to code production, yet may improve code development processes, we suggest that the RP perspective may be integrated naturally in XP. In the description that follows, readers' familiarity with the XP practices is not necessary.

Tables 1 and 2 present two ladders of reflection[1]. The first one (Table 1) presents a ladder of reflection that is constructed during a pair programming[2] session. The second ladder (Table 2) is constructed during a Planning Game session.[3] In the first case the idea is to illustrate how a pair of programmers may increase the abstraction level of their thinking when reflection is interwoven within the process of software development. In the second case we illustrate how the fact that the customer

[1] Tables 1 and 2 illustrate the participants' thinking/discourse at each reflection-rung.

[2] The practice of pair programming specifies that any piece of code should be written by two developers, each of whom has a different role: the one with the keyboard and the mouse thinks about the best way to implement a specific task; the other partner thinks more strategically. As the two individuals in the pair think at different levels of abstraction, the same task is thought about at two different levels of abstraction *at the same time.*

[3] The planning game defines a process in which the customer, together with the developers, defines his/her requirements and priorities. One of the significant advantages of the planning game is that both the customer and the entire team participate in it, and thus all know the development process. Furthermore, guidelines that lead to decisions with respect to a specific release or iteration are clear to all.

and the team define together the next release/iteration makes it possible to introduce a reflective mode of thinking.

Table 1: A ladder of reflection: The case of pair programming

Ladder rungs	***Pair dialogue***
Designing *[a process of reflection-in-action]*	A: Did we consider all the exceptions?
Description of designing *[it takes the form of description with: appreciations, advice, criticism, etc.]*	B: Good question. Let's think about the best way to search for exceptions. I'm trying to understand what to think about when I'm looking for potential exceptions.
Reflection on description of designing *[reflection on the meaning the other has constructed for a description he or she has given]*	A: I think that this is not such a simple task. I have never thought about such systematic ways to look for exceptions. OK. Let's give it some thought. [*Working on formulating a systematic way for finding exceptions*]
Reflection on reflection on description of designing *[the parties to the dialogue reflect on the dialogue itself]*	B: Now that we have developed a systematic way for finding exceptions, I think we must analyze these strategies and reflect on the path that led us to finding these guidelines. A: Yes, this may improve our ability to solve problems of a similar nature in the future.

Table 2: A ladder of reflection: A Planning Game session

Ladder rungs	**A conversation during a planning game session**
Designing *[a process of reflection-in-action]*	Customer: In fact, I want this feature to behave this way [*moves her hands to illustrate*]. Developer 1: Can you think about a similar feature you needed before?
Description of designing *[it takes the form of description with: appreciations, advice, criticism, etc.]*	Customer: What do you mean? Would you like me to think about a similar case in the past in which I wanted a similar feature? Interesting. I have never been asked to do something like this before. But yes, I can think about a situation in the past when we needed a new system for our inventory management. I wanted the application to have this feature and only when we received the system I realized that, in fact, what we need is something else, more … [*illustrates with her hands*]. Let's call it B. Wow! Does that mean that we should not have at all the feature I described before?
Reflection on description of designing *[reflection on the meaning the other has constructed for a description he or she has given]*	Developer 2: We do not know. We can check the two options. But, can you please recall, what, in the case you just mentioned, led you at the end to realize that what you need is B, and why you didn't (or couldn't) realize this before, I mean, before you got the system and started working with it. Customer: Truly, the problem was that we did not consider the full setting in which the system would work. I think that we should consider the same issue now, before I make the final decision. [*The customer and the developers think about the way the application will be used, focusing on the specific considerations that were neglected in the customer's previous experience. At the end they decided about a third option that should be applied for these specific circumstances.*]

Reflection on reflection on description of designing *[the parties to the dialogue reflect on the dialogue itself]*	Customer: It's amazing. I must trace with you the full path we went through together. [*The customer and developers dedicate the next 15 minutes for this purpose*]. Customer: I do not want even to think about the catastrophe that could have happened if you develop one of the first two options we talked about. I must learn the lesson. First of all, I'd like to apologize for my resistance to take part in the planning game. I must confess that only now I understand how I should manage the all business with the new application. Tracker: I think we also learnt something from this experience. First, we should not be afraid to ask our customers difficult questions and to insist on getting answers. Second, the specific circumstances you introduced us to may be useful in our future projects. Finally, we should remember that before making final decisions and moving on, sometimes it is worth checking whether we consider all options. I believe that eventually, even if we stay with the first option, this would not be considered a waste of time.

Looking at the various rows of Tables 1 and 2, one may find that the subjects of reflection on each rung are objects of different levels of abstraction: While detailed elements are the focus on the first rung, ways of thinking and heuristics are at the center of attention on the fourth rung. As can be observed, the participants improve their understanding throughout the scenarios described in Tables 1 and 2.

5. Conclusion

This position paper describes a framework for adopting an RP perspective in general and the construction of ladders of reflection in particular into software engineering processes. Specifically, we illustrate how the awareness to the potential contribution of ladders of reflection to software development processes may improve developers as well as customers' understanding of processes they are engaged in.

We propose to discuss with the workshop participants the following questions:

- How to identify situations in software engineering in which a reflective mode of thinking in general and a construction of ladders of reflection in particular may be suitable.
- How to assimilate a reflective mode of thinking into these situations for which it is identified that a reflective mode of thinking may contribute.
- How to educate software developers to efficiently construct ladders of reflections.

6. References

Beck, K.(2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

Schön, D. A. (1983). *The Reflective Practitioner*. BasicBooks,

Schön, D. A. (1987). *Educating the Reflective Practitioner: Towards a New Design for Teaching and Learning in The Profession*. San Francisco: Jossey-Bass.

On the Timing of Presenting Situational Talkback in Support of Reflective Practitioners

Yunwen Ye[1,2]

[1]Center for LifeLong Learning and Design
Department of Computer Science
University of Colorado, Boulder, CO80309-0430
+1 303 492 3547

[2]SRA Key Technology Laboratory
3-12 Yotsuya, Shinjuku
Tokyo 160-0004, Japan
+1 81 3 3347 9361

yunwen@cs.colorado.edu

INTRODUCTION

Design is an activity of making plans to inform the process of making, in order to create artifacts we want to surround us [5]or change the current situation to our satisfaction [9]. Since the primary goal is to inform making, certain representations are bound to be produced during the process of designing. Design could be logically regarded as the repetition of making the representation (action), and interpreting the partially finished representation (reflection) in terms of its fitness with the ideal image in the mind of the designer. The interpretation leads to further actions that either change the current representations for better fit or add more representations for better approximation of the ultimate design goal.

If we view the design process as a goal-oriented cognitive process in which the goal itself is sometimes not well defined, each action during the design process could be viewed as a decision-making one which is chosen from possible candidates after deliberation. The decision-making process, is dynamically determined by the knowledge that the designer has in his or her mind and the information presented in the workspace in which the designer is placed. In other words, the design process is a continuous dialogue between human minds and the interim representations [8] Designers act to make representations, reflect upon the information presented by the representations, and act further after reflection.

Because the information contained in the workspace and the representation is an important resource for reflection, the important research questions for providing support for reflective practitioners are: What information should be presented? How should such information be presented? And when should such information be presented? This position paper describes some of my research efforts in understanding the timing of presenting information in support of reflective designer.

THE TIMING OF PRESENTING INFORMATION

Design is a knowledge intensive activity and result from the design activity is a knowledge artifact that embodies the knowledge of the designer. The knowledge that comes from the designer could be acquired by the designer through three different phases: before, in and after the action.

The knowledge that has been acquired by the designer *before* he or she starts design is the result of his or her professional education and experience. In other words, the knowledge is the result of previous learning or interpretation of information presented to the designer long before the design starts. The context in which learning takes place is different from the context in which the learned knowledge is applied.

The knowledge acquired *after* the action is finished is a process of learning from previous experience, from the feedback information of the action. To support reflective designers to reflect on their finished action, feedback information is presented when the action for which the information is provided has been finished. Feedback can create a situational backtalk of the action by pointing out a potential breakdown the designer has not known or noticed, or can augment the situational backtalk to help designers reflect better on the action just completed. Feedback can serve two roles. First, it creates a learning opportunity for designers to improve work performance. For example, the ACTIVIST system [4] teaches users the corresponding key shortcut to replace a series of complex keystrokes used in their previous action in a text editor. Second, if the previous problematic action can be undone or modified, it helps users reach a better solution, such as the on-the-fly spell-checking mechanism in many word-processing systems.

The knowledge acquired *in* the action is a process of expanding the knowing-in-action. For each design situation, there is a period of time called *action-present* in which the designer remains in the "same situation." This is a period of time that the designer has made up his or her plan of action but has not executed the necessary operations to change the situation. Information presented in the action-present period could be immediately acquired by the designer and applied to change the designer's original design plan. Information presented in this period of time is feedforward [9] information because it can make designers change the course of action or assist designers in accomplishing the action [10].

These three forms of knowledge acquisition could be unified in terms of the temporal relationship between the timing of the information being presented and the timing of its application. In the first case, information is presented without any advanced knowledge of its potential context. In the second case, information is presented when the context is still remembered or can be easily reconstructed by the designers. In the third case, the information is presented right into its application context.

OPPORTUNITIES FOR SUPPORTING REFLECTION-IN-ACTION IN SOFTWARE DEVELOPMENT

Software development involves many design activities. Software developers engage in many reflections-in-action. However, few systems have been developed to provide explicit support for reflective software developers. I want to explore the opportunities of supporting reflection-in-action by applying the framework described in the previous section.

The goal of software development is to create computational representations for problems to be solved. Since the representations are computational already, some might think that software should be the best domain for augmenting situational talkback to support reflection-in-action. Apparently, this is not the case. Except for a few exceptions [3, 7], very little software engineering research literature has brought up this issue. However, a careful analysis of certain efforts in software engineering leads to the conclusion that the research community has, all the time, been trying to do that implicitly.

Brooks claimed that the invention of interaction programming environment is the most significant development in advancing our capability of software construction [2]. If we analyze this observation from the perspective of reflection-in-action, we can see it is simply because the representational feedback information is presented to the programmer much closer to its original context, in terms of timing. It is well accepted that interpretative programming language, though less efficient performance-wise, is easier to use and easier to learn because programmers can try their programs immediately without going through the save-file and compilation phase.

Code review is a process in which project team come together to review the code written by one member to find bugs in the code or provide feedback to improve the code. The representational talkback comes from the social environment. Due to the difficulty of organizing reviewing team and coordinating review process, program review is often an expensive process and cannot be done right after the code is written or at the needs of the software developer. Computationally networked socio-technical environment provides a new opportunity of coordinating such social process of providing representation talkback. For example, one of the success factors in Open Source Software development is that "given enough eyeballs, all bugs are shallow [6]." Support for reflective software developers means not only providing computational mechanisms that augmenting or presenting timely the situational talkback, but also facilitating the timely presentation of socially situational talkback.

Rapid prototyping, especially research on executable specification, is yet another effort. There is a long separation between the phase in requirement acquisition (problem framing) and the development of executable code (problem solution) that can provide situational talkback. To make the initial specification executable is an effort of shorten the time separation so that modification could be made earlier and easier.

Extreme programming and agile development methodology [1] has pushed this a little further by breaking down the long separation between the framing of problems and the solution of problems. Functionality is added incrementally as a result of incorporating the feedback from the users. Pair programming is an effort of providing immediate socially representational talkback.

There are many tools developed for analyzing ripple effects or test coverage or slicing. All those tools are supposed to provide feedback information on certain software development actions. For example, ripple effect analysis is meant to identify the range of code that is affected by code or design modification. However, the cycle of making modification and getting the feedback is too long and too cumbersome because the analysis is only available when the modification is made final. Software developers can get better tool support if the situational talkback of the modification is presented immediately after the modification is made. If the feedback is presented even before the code modification is finally committed, software developers can execute reflection-in-action better.

The above list is definitely limited, but serves as a starting point of thinking how to support reflective software developers with better development methodology and tools that presents representational backtalk in a timely fashion.

CONCLUSION

This position paper describes my research efforts in reframing some research problems in software development from the perspective of the theory of reflection-in-action, and in exploring the opportunities of supporting reflective software developers. Most of the thinking remains theoretical and will be evaluated further in the near future with system implementation and empirical studies.

REFERENCES

1. Beck, K. Extreme Programming Explained. Addison-Wesley: Reading, MA, 2000.

2. Brooks, F.P.J. The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary edition. Addison-Wesley: Reading, MA, 1995.

3. Fischer, G. Domain-Oriented Design Environments. Automated Software Engineering, 1994. 1(2):177-203.

4. Fischer, G., Lemke, A.C., and Schwab, T. Knowledge-Based Help Systems, in Human Factors in Computing Systems (CHI'85): San Francisco, CA, 1985, 161-167.

5. Habraken, N.J. The Appearance of the Form: Four Essays on the Position Designing Takes Between People and Things. Awater Press: Cambridge, MA, 1985.

6. Raymond, E.S. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly: Sebastopol, CA, 2001.

7. Robbins, J.E., and Redmiles, D.F. Software Architecture Critics in the Argo Design Environment. Knowledge-Based Systems, 1998. 11:47-60.

8. Schön, D.A. The Reflective Practitioner: How Professionals Think in Action. Basic Books: New York, 1983.

9. Simon, H.A. The Sciences of the Artificial, Third edition. The MIT Press: Cambridge, MA, 1996.

10. Ye, Y., and Fischer, G. Supporting Reuse by Delivering Task-Relevant and Personalized Information, in Proceedings of 2002 International Conference on Software Engineering (ICSE'02) (Orlando, FL., 2002), 513-523.

Interfaces that Reduce the Cost of Examining Alternatives

Position paper for CHI 2004 Workshop on Designing for Reflective Practitioners.

Aran Lunzer
University of Copenhagen, Denmark
aran@bigfoot.com

Introduction

My contribution to supporting reflection in the use of computer tools lies in designing cognitively low-cost mechanisms for requesting and working with many scenarios simultaneously. Rather than being stuck with an interface that allows manipulation of just one case at a time, a user can bring many cases together, side by side, and is thus helped in reflecting on the relationships between them.

The work reported here began as my PhD project in the early 1990s. The motivation was the observation that many computer applications (including search engines, document formatters, design tools), though offering great flexibility in the range of results they could deliver, made it hard for users to request and compare different results. Consequently, users of such tools were motivated to satisfice, i.e., to accept the first result that appeared acceptable, rather than to explore further to see whether alternative scenarios may produce results that were better. I felt that if the cost of making such explorations could be reduced sufficiently, this would tip the balance in favour of exploration, and hence lead to better-informed decisions.

This paper summarises the developments and research directions arising from that original motivation.

Result-space reconnaissance

The focus of my PhD (Lunzer, 1996) was the idea of 'result-space reconnaissance' – giving the user a cheap way to examine a wide range of scenarios, by dispatching the computer to evaluate them and to report just summary values that help with evaluation and comparison. The key features of a result-space reconnaissance setup are therefore as follows:

- A user specifies the scenarios that are to be examined – in terms of alternative settings for various input parameters – and the measurements to be taken from each scenario.
- The computer evaluates all specified scenarios, makes the requested measurements on their results, and provides an interactive visualisation that correlates the parameter settings with the measurement values.
- The user works with the visualisation to find scenarios with interesting combinations of properties, and to request the detailed results for those scenarios.

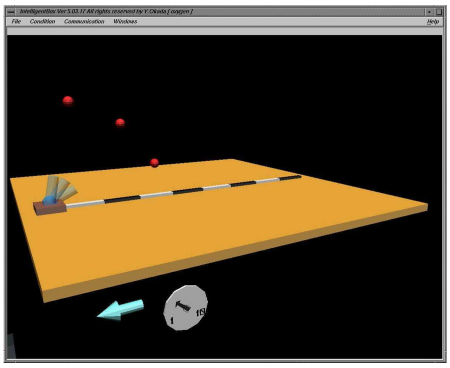
The chosen domain of study was the formatting of documents in LaTeX, given freedom over such parameters as base font size, line spacing, the scaling of embedded figures, and so on. For what could be argued was a small additional investment in effort, users could have their documents formatted in dozens of alternative ways, and could discover which results had desirable properties such as fitting onto a certain number of sheets, or having the page breaks occur at what the user felt to be convenient places in the text.

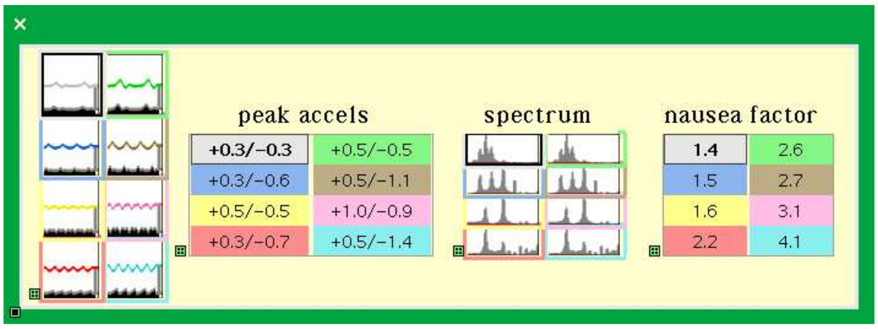
The outcome was mixed. Although some test subjects felt that this was a worthwhile trade-off of effort against the potential to uncover excellent formatting results, others were less motivated. Some felt that any output of LaTeX was probably good enough; to borrow the words of Gerhard Fischer (2002), they simply wanted to be consumers. Furthermore, the extra effort was spent on using the mechanisms for requesting reconnaissance (pseudo-LaTeX markup), and the interactive parallel-coordinates visualisation of the results – both of which are distractions from a user's core activity of preparing a document. This signalled the need for a more integrated approach to handling alternatives, embedded within the main tool for the supported activity.

Subjunctive interfaces

The 'subjunctive interface' approach (Lunzer 1998, 1999; Lunzer & Hornbæk 2003) was inspired by Hofstadter's (1979) idea of a magical television that could show a viewer not just the broadcasts from a single reality, but from alternative realities that differed in ways freely chosen by the viewer – such as realities involving different weather conditions, or taking place on entirely different planets. The key features of a subjunctive interface are as follows:

- The user of an application should be able to set up multiple scenarios that differ in arbitrary ways. For example, when offered some choice in the application's interface, the user should be able to say 'maybe I want value X, but maybe Y or Z... so let me set up all three and see how things turn out in each case'.
- The scenarios should be viewable simultaneously, side by side, in a way that helps the user to compare them and also to keep track of all the values – all the inputs, and their corresponding results – belonging to each scenario. The results should be shown using essentially the same display techniques normally used for the single-scenario version of the application.





peak accels	
+0.3/−0.3	+0.5/−0.5
+0.3/−0.6	+0.5/−1.1
+0.5/−0.5	+1.0/−0.9
+0.3/−0.7	+0.5/−1.4

nausea factor	
1.4	2.6
1.5	2.7
1.6	3.1
2.2	4.1

Figure 1: Left: An early example of a subjunctive interface, applied to a model of the flight of projectiles launched against a headwind. In the situation shown here, the user has chosen to create multiple scenarios based on different values for the angle of the launcher – specified by pressing a hot-key at various points during the direct-manipulation adjustment of that angle. As a result, three launchers now exist in parallel; the three balls seen here in mid-flight were launched simultaneously in the three scenarios a few seconds earlier. The user can now adjust wind speed (using the large arrow at bottom left) and projectile mass (the dial) and observe simultaneously how the launches from the different angles would be affected. If wanted, these parameters too can each be given multiple values; this will simply multiply the number of balls launched on each firing.
Right: A feedback tool for a vehicle designer, showing the simulated response of a suspension system running over a bumpy road. The designer has control over a number of variables in the suspension design (not shown in this figure), and is here examining how the current settings would perform in eight scenarios representing alternative road patterns and speeds. Each of the four response measurements (motion plot, acceleration values, frequency spectrum, and a motion-sickness measure) is showing dynamically updated values for each of the scenarios, arranged as 'small multiples' displays to help the user correlate the values between them. For example, the scenario at top left would subject passengers to accelerations of ±0.3, and has an estimated nausea factor of 1.4.

- The user should be able to control scenarios in parallel, for example by adjusting an input parameter that is shared by many scenarios and seeing instantaneously the effect of this adjustment on each scenario. (One could say that the cues available for reflection-in-action during such adjustment are thus multiplied by the number of scenarios being affected.)

Two examples of the many experimental subjunctive interfaces that have been built to date are shown in Figure 1. These are examples of design based on dynamic simulation. Subjunctive interfaces can also be useful for design of static artefacts – such as an architectural model that incorporates what-if elements, or a year planner that allows each future event to be scheduled for several alternative possible dates. In these latter domains the interface would help the user to work with the branching possibilities established by those alternatives, for example offering rich reflection-in-action when the user comes to add further elements and discovers what would be (in)consistent with the existing elements in each scenario.

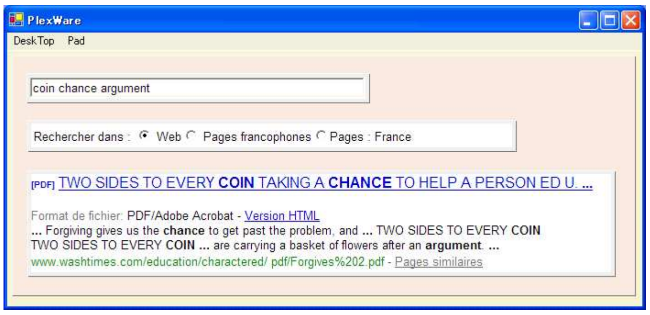
As well as some user studies in Copenhagen demonstrating significant benefits from a subjunctive interface designed for browsing and comparing multi-attribute data (Lunzer & Hornbæk, to appear), other researchers have begun to obtain successful results from their own extensions of the theme – notably the recent work on supporting alternatives in image processing, under the banner Parallel Paths (Terry *et al.*, to appear).

Future: Ingredient-Based Computing

Figure 2 shows the first, simple example of a tool demonstrating Ingredient-Based Computing – a specialisation of subjunctive interfaces suited to applications that can be characterised as spreadsheet-like acyclic graphs of cells and formulas. This application model, which we believe can subsume a wide range of today's applications (including, not least, spreadsheets), offers opportunities to simplify the handling of knock-on effects of alternative values that the user assigns to cells within the dependency graph.

One goal for ingredient-based computing is to cater for mixed-initiative variation – i.e., to allow alternatives to be generated not by direct user request but from automated processes. For example, computational processes that provide unsolicited information for user reflection (such as critics, context-sensing recommenders, or other forms of agent) may come up with conflicting values depending on the assumptions that each embodies. Rather than try to muddle all these elements together, it would make sense to replicate scenarios automatically in a way that lets the user understand the various offerings – including how each has arisen – and make informed choices about which to attend to.

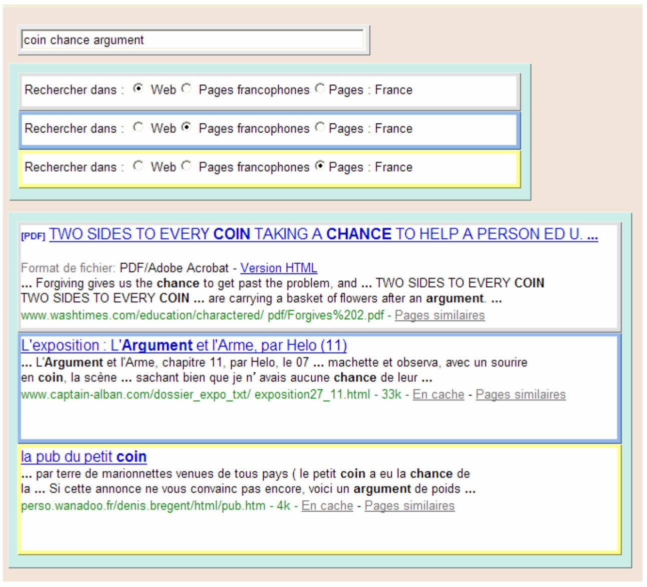
My hopes for this workshop include finding an opportunity to discuss concrete issues relating to such mixed-initiative support, and also relating to situations that would call for establishing long-lived alternatives – i.e., scenarios that remain separate, yet individually relevant, for weeks or months rather than just during the course of a single usage session.



coin chance argument

Rechercher dans : [✓ Web] Pages francophones Pages : France

[PDF] TWO SIDES TO EVERY **COIN** TAKING A **CHANCE** TO HELP A PERSON ED U. ...
Format de fichier: PDF/Adobe Acrobat - Version HTML
... Forgiving gives us the **chance** to get past the problem, and ... TWO SIDES TO EVERY COIN
TWO SIDES TO EVERY **COIN** ... are carrying a basket of flowers after an **argument**. ...
www.washtimes.com/education/charactered/ pdf/Forgives%202.pdf - Pages similaires



coin chance argument

Rechercher dans : [✓ Web] Pages francophones Pages : France

Rechercher dans : Web [✓ Pages francophones] Pages : France

Rechercher dans : Web Pages francophones [✓ Pages : France]

[PDF] TWO SIDES TO EVERY **COIN** TAKING A **CHANCE** TO HELP A PERSON ED U. ...
Format de fichier: PDF/Adobe Acrobat - Version HTML
... Forgiving gives us the **chance** to get past the problem, and ... TWO SIDES TO EVERY COIN
TWO SIDES TO EVERY **COIN** ... are carrying a basket of flowers after an **argument**. ...
www.washtimes.com/education/charactered/ pdf/Forgives%202.pdf - Pages similaires

L'exposition : L'**Argument** et l'Arme, par Helo (11)
... L'**Argument** et l'Arme, chapitre 11, par Helo, le 07 ... machette et observa, avec un sourire
en **coin**, la scène ... sachant bien que je n' avais aucune **chance** de leur ...
www.captain-alban.com/dossier_expo_txt/ exposition27_11.html - 33k - En cache - Pages similaires

la pub du petit **coin**
... par terre de marionnettes venues de tous pays (le petit **coin** a eu la **chance** de
la ... Si cette annonce ne vous convainc pas encore, voici un **argument** de poids ...
perso.wanadoo.fr/denis.bregent/html/pub.htm - 4k - En cache - Pages similaires

Figure 2: A trivial example of ingredient-based computing, showing the pursuit of multiple parallel navigations on the World Wide Web. On the left the user has set up a combination of cells in a tool called C3W (Clip, Connect and Clone for the Web) that is being developed in collaboration with Jun Fujima at Hokkaido University [paper submitted for review]. These cells contain active elements extracted from the search site google.fr, and hande, respectively, the search input field; the switch that selects whether the search should cover the whole Web, just pages in French, or just pages in France; and the top result (showing page name and keyword context) from a completed Google search. On the right the user has replicated the switch cell to create three alternatives, and has set each to a different option; because the search-result cell depends directly on the switch (through a pseudo-formula whose 'calculation' involves submitting a query to Google), it has been automatically replicated too. Entering new search keywords into the tool in this state will launch three queries, and display the three respective top results.

References

Fischer, G. (2002) 'Beyond "Couch Potatoes": From Consumers to Designers and Active Contributors.' *First Monday* **7**(12), December 2002.
`http://www.firstmonday.dk/issues/issue7_12/`

Hofstadter, D.R. (1979) 'Gödel, Escher, Bach: an Eternal Golden Braid'. Basic Books.

Lunzer, A. (1996) *Reconnaissance: a widely applicable approach encouraging well-informed choices in computer-based tasks.* Ph.D. thesis. TR–1996–4, Dept of Computing Science, Univ of Glasgow, Feb 1996. 266pp.

Lunzer, A. (1998) 'Towards the Subjunctive Interface: General Support for Parameter Exploration by Overlaying Alternative Application States.' In *Late Breaking Hot Topics Proceedings of IEEE Visualization '98,* Research Triangle Park, North Carolina, Oct 1998, 45–48.

Lunzer, A. (1999) 'Choice and Comparison Where the User Wants Them: Subjunctive Interfaces for Computer-Supported Exploration.' In *Proceedings of IFIP TC. 13 International Conference on Human-Computer Interaction (INTERACT '99),* Edinburgh, Scotland, Aug 1999, 474–482.

Lunzer, A. and Hornbæk, K. (2003) 'Side-By-Side Display and Control of Multiple Scenarios: Subjunctive Interfaces for Exploring Multi-Attribute Data.' In *Proceedings of the Australian Conference on Computer-Human Interaction (OZCHI 2003),* Brisbane, Australia, Dec 2003, 202–210.

Lunzer, A. and Hornbæk, K. (to appear) 'Usability Studies on a Visualisation for Parallel Display and Control of Alternative Scenarios.' In *Proceedings of The 7th International Working Conference on Advanced Visual Interfaces (AVI 2004),* Gallipoli, Italy, May 2004.

Terry, M., Mynatt, E., Nakakoji, K. and Yamamoto, Y. (to appear) 'Variation in Element and Action: Supporting Simultaneous Development of Alternative Solutions.' In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2004),* Vienna, Austria, April 2004.

Notes