Proceedings of the International Conference on Knowledge Capture – Banff, Canada – October 2005

Knowledge Management for Product Maturity

Guy A. Boy
European Institute of Cognitive Sciences and Engineering (EURISCO International)
4 avenue Edouard Belin, Toulouse, France
email: guy.boy@eurisco.org

ABSTRACT
When a new product is delivered, it seldom meets all customer needs. The mature phase of a product is driven by customer needs. It requires a human-centered development cycle. As a result, the company should be able to listen the voice of its customers. Most industrial companies are driven by engineers and by technology itself. If current technology is to serve all actors of the life cycle of a product, related companies need to change their ways of dealing with maturity. They have to stop being so driven by features and start examining what customers actually do. The concept of customer itself has to be revisited to the point that any person or group who deals with a product (coming from a process) is a customer of those who developed the product. Product maturity and process maturity are usually distinguished. Product maturity is related to end-user satisfaction, i.e., customers. Product maturity deals with user experience. Process maturity is related to designers, developers, maintainers and other actors who have an impact on the making and evolution of the product. Process maturity deals with organizations, communities and teams involved in the production of a product. This paper proposes an integrated approach to product and process maturity that involves the use of active design documents to support the description of what the product is, how it is or should be used, why it is designed the way it is and how much it will cost to customers in terms of performance, safety, comfort and other criteria that may be relevant to the product purpose of use.

General Terms
Design, Human factors

Categories and Subject Descriptors
I.2.1 Applications and Expert Systems (H.4 Information Systems Applications); I.2.4 Knowledge Representation Formalisms and Methods – *representation languages, semantic networks.* I.2.11 Distributed Artificial Intelligence; J.6 Computer-Aided Engineering.
See http://www.acm.org/class/1998/overview.html

Keywords
Maturity, knowledge management, user experience, design and development, active design documents.

INTRODUCTION
The design of software-intensive systems is still a question of expertise distributed among a large variety of actors requiring a well-orchestrated organization. The design of large aircraft for example involves several types of experts. Today, taking into account user requirements has become a crucial issue in order to satisfy criteria such as safety, comfort, learnability, performance, and traceability of product design/development and modifications decisions. Therefore, industry must manage more consistently knowledge that is required for certification, training, maintenance and/or operations (use). This emerging industrial need is expected to lead to global quality by taking into account user experience at all critical stages of the life-cycle of a product. This is the rationale of a novel approach in enterprise integration that takes pragmatics acquisition seriously. This paper presents the main concepts, methods and tools that support this approach to human-centered development of a product.

Pragmatics acquisition still requires a better understanding of the notion of *context*. Contextualization of knowledge is not a novel issue. As an example, we have already shown the relevance and usefulness of active design documents (ADDs) [5] for human-centered design in aerospace. The integration of ADDs within the group elicitation method (GEM) [3] offers a mediation space facilitating pragmatics acquisition, cooperation and coordination among actors of the life-cycle of a product.

This paper first discusses the need for an integrated sociotechnical approach to product and process maturity. The claim is that maturity is a matter of controlled information management involving appropriate organizational models. Pragmatics will be presented in the framework of knowledge management (KM). ADDs will be described as methodological tools to support pragmatics in the management of product attribute descriptions. ADDs force the contextualization of product knowledge along with the product life-cycle. Decision-points traceability emerges from the incremental contextualization of product attributes. The balance of the paper is devoted to a discussion on ADDs possibilities and perspectives.

SOCIOTECHNICAL MATURITY
A technology becomes mature when it is useful, usable and acceptable, i.e., is socially accepted, meets legal requirements, and answers relevant commercial issues. User experience enhances technology-centered maturity approaches. Users are all actors involved in the life cycle of the product, e.g., end-users, customers, maintainers, trainers, and designers. In the early stages of a technology, products are driven by the needs of technically sophisticated consumers.

The operational life-cycle of a technology can be divided into two periods that are characterized by different maturity criteria: (period 1) technology and performance; (period 2)

ease of use, reliability and price. User behavior changes [14]. For example, in the early stages of computer industry development in the 70s, computers were big and mostly used by highly-skilled engineers. Computer use was a matter of technical performance. Microcomputers emerged and democratized the use of information technology to the point that most people have a computer at home today. Microcomputers arrived during the first half of the 80s. Many engineers at that time did not want to use such a new technology because they thought that it was made for technically low-skilled users. The transition from period 1 to period 2 was being reached at that point. Today, computers are becoming invisible, integrated within the most familiar tools such as the telephone, automobile, or microwave. This is another transition point.

At such transition points, maturity is an issue. The best way to master maturity is to improve the period 0 that includes design and development of the product. The main issue here is that it is very difficult and almost impossible to predict the future without relevant data. Experience feedback and expert knowledge are often required to make appropriate design and development decisions. Instead of periods 0, 1 and 2, it is much better to work on periods n, n+1 and n+2. This assumes that we work on a family of products. This product family issue is crucial and has emerged for many industrial products including aerospace, software and telecommunication. Thus n-1 knowledge is incrementally used in period n.

MATURITY REQUIRES APPROPRIATE INFORMATION MANAGEMENT

Product maturity results from the culture of the organization handling the product from design to manufacturing to operations. Sociotechnical maturity is often a matter of appropriate organizational interactions that include good communication, cooperation, coordination, supervision and participatory design. These emerging processes are strongly influenced by new requirements coming from time compression, global quality assurance, and insightful articulation work. New profiles of managers are required including (music-like) conductors and interaction architects.

Three types of organizational interactions among agents are considered: (1) supervision; (2) mediation; (3) cooperation by mutual understanding [7]. In addition, organizations have moved from an army-type model to orchestra-type model. In the former, the organization is hierarchical and the maturity of the structure is crucial. It is interesting to note that ISO standards were developed with such an organizational model in mind. In the latter, the organization is a network of experts that may be individuals, groups, organizations or communities. The maturity is in the various interactions among the experts. The metaphor of the orchestra is appropriate since each musician is individually skilled for a given type of instrument. However, the symphony will not be a good product without an appropriate music theory that they share, and a conductor who coordinate the orchestra. We are looking for such a music theory.

Product maturity results from many factors that are sometimes very difficult to identify and master. People who design a new product must be visionaries. They are motivated and usually look for the unknown. They have to make decisions, sometimes-crucial decisions and in large industrial programs, group decisions. Group work has then to be well supported. Decisions lead to new designs and continuous validation. Maturity is always the results of proactive investigations. This paper presents a solution to this end.

PRAGMATICS IN KNOWLEDGE MANAGEMENT (KM)

Morris seems to be the creator of *pragmatics* in 1938. Three branches of semiotics are usually distinguished: *syntax* that studies formal relations between signs; *semantics* that studies relations between signs and objects that signs represent; and *pragmatics* that studies relations between signs and their interpretations or uses [10]. According to Francis Jacques, « pragmatics studies language as a discursive, communicative and social phenomenon » [1]. In this paper, pragmatics is limited to its potential application in KM. The integration of signs is an integral part of KM. Signs are also called terms that represent concepts. In a specific domain, *terminology* defines a set of terms, instead *ontology* defines a set of concepts. A first difficulty in this rationalization enterprise is that technology often evolves faster than research can handle. Another difficulty is that there are a variety of research fields that barely communicate with each other such as ethnography, human factors, philosophy, information and cognitive sciences.

Knowledge management is a matter of production and regulation of knowledge; both processes involve contextualization. Homeostasis that comes from the theory of cybernetics enables the study of information regulation [17] [18]. Autopoiesis enables the study of knowledge production [12]. In our KM approach, pragmatics acquisition is performed by incrementally contextualizing knowledge. In that sense, knowledge is auto-reproduced according to its own use. Contextualization of knowledge consists in making (more) sense and therefore contributes to the reconciliation of two major philosophical trends that are positivism and phenomenology. Tuomi [16] proposed the 5A model for knowledge generation. This model has five major functions: anticipation, appropriation, articulation, accumulation and action. Knowledge generation is seldom possible without action. For example, we learn from differences between the anticipated behavior of our environment and the effectively perceived situation. This process is sometimes denoted as discovery. Existing knowledge chunks can be articulated and reshaped in order to generate new knowledge. Knowledge *accumulates* in a memory following an iterative process similar to Piaget's accommodation mechanism.

All organizational-learning-related processes work when the organization is well articulated. Knowledge accumulation is not a pure addition of knowledge chunks, e.g., an addition of new rules for example. It is a transformation of existing knowledge and its interrelations. Of course, appropriate formalisms are required to develop and maintain a computer support to KM. These formalisms should be interoperable and enable knowledge sharing among actors. They may have the drawback to bias the initial meaning of represented knowledge and thus to reduce its outreach. In this paper, formalisms being used are hypertext and knowledge blocks that have been renamed interaction blocks [6]. Interaction blocks were developed blocs to model the use of operational procedures in the aerospace domain. The block formalism was used to model context attached to hyperlinks of a document network. Blocks include two types of conditions: contextual conditions that provide positive experience and abnormal conditions that provide negative experience. The underlying contextualization process was extensively developed in the *Computer Integrated Documentation* (CID) project developed at NASA [2] for a very large documentation corpus. The current work presented here is based on this approach.

DOCUMENTATING THE DESIGN PROCESS AND ITS SOLUTION

Three types of short-term perspectives emerge [4].

- *Improving document generation*. Documentation is difficult to do during the design process itself; it is usually done under pressure. It is observed that people tend to reuse document templates that they successfully used in the past. It appears that documentation management should be more structured in a case-based manner. In addition, cooperative generation of documents should be enhanced. Documents are mediating tools that enhance (or don't if they are badly designed) communication between design team members.
- *Motivation and skills*. It is not usually fun to write about what you have already designed. Most designers do not like to write. This is a matter of culture and education. Designers should be trained to write. Documentation authoring tools should facilitate the development of documents.
- *Design knowledge reuse and evaluation (legal issues).* Design decisions often need to be retrieved to explain why a piece of equipment was designed the way it was and not another way. There are two types of motivations: one is design knowledge reuse and another is evaluation/certification. The first one is related to the development of organizational memory systems to improve corporate efficiency and competence continuity. The second motivation is related to legal issues and usability criteria used during the various stages of the decision making process in design.

In the middle term, documentation should be more imbedded within the design process. Rapid prototyping and multimedia documentation should be further investigated together. This could help both design decisions and dynamically document them. In particular, the concept of an active document appears to be very interesting since it includes both aspects. Design teams should be trained on design documentation requirements and development. At least technical writers should be involved in design teams. It would seem that an appropriate use of technical documentation during design would tend to redefine design jobs. Design and writing activities should be considered as similar activities that should support each other. The traditional structuring/indexing/information-retrieval issue is still very relevant. Knowing that a useful piece of information is available somewhere is crucial when taking into account the speed of change in technology. How can information pieces be linked in context? Characterization of context is very difficult and needs further deeper investigations.

Among the key issues are:

- Design rationale; implementation of semi-formal frameworks for design decisions; knowledge representations; formalization of different perspectives;
- Passive and active documentation (e.g., prototypes); relations between formal and informal; use of scenarios; relations between physical and virtual (hybrid objects);
- Documentation for validation; improvement of usability tests of organizational memories; implementation of guides and recommendations;
- Organizational roles of documentation; documentation for structuring the design process; design knowledge reuse; documentation generated under temporal constraints; remote design work;
- Ease of creation and use of documentation; integrated design-environments; visualization of the design process; design and implementation tools; metaphors that designers use to speak about design; documentation for requirement engineering;
- Documentation for communication among design team members; transparency of the design process; documentation for communication with customers; documentation for training.

The work presented in this paper claims that the quality of a technical documentation contribute to the quality of design and development. We write for potential readers. Similarly, we design for potential users. We know that papers that we write need to be reviewed by several people before being distributed. We also know that products need to be tested by several people before being distributed. Readers of multimedia documents have become users of software applications. From this viewpoint, reading has evolved toward human-computer interaction (HCI). Writing has also evolved toward interactive software design. Writing words, sentences, paragraphs and chapters evolves toward design-

ing objects and software agents [8]. Static paper documents become (inter)active documents.

ACTIVE DESIGN DOCUMENTS

During the life cycle of a technical documentation, design teams write requirement documents for development teams that in turn write documents for potential users. Operational documentation is usually designed when the product is developed. Unfortunately, it often contributes to compensate design flaws. Generally, design documents describe the way products work. Documenting the design process and its solution is or should be an integrated tool serving human-centered design and traceability.

The active part of a book (system) is the reader (user). In addition, the organization of the book (system), the way sentences (objects) are written (designed), used style and lexicon suggest a specific activity from the reader (user). Sometimes, the reader (user) hardly understands what the author (designer) wanted to express. Instead of mobilizing reader's (user's) cognition on interaction issues, the most important part of reader's (user's) cognitive activity should be mobilized on understanding and interpretation of (active) document content.

Taking into account user requirements in the design/evaluation process led to the development of human-centered design methods. Instead of designing a product and documenting it after, it is better to design and evaluate documented prototypes, called *active design documents* (ADDs) [5], incrementally until they become acceptable prototypes. One of the main difficulties in the design of technical documents is to anticipate a large number of contexts of use. The context of use is related to other entities such as situations, behaviors, viewpoints or dialog. It is extremely difficult to contextualize by using conventional paper technology. The use of software technologies provides more contextualization possibilities. Contextualization is not only a document-intrinsic issue, but also a document-extrinsic issue related the environment of the document (traceability).

An ADD is a hypermedia application usable by a community of people. An ADD describes various attributes of a product (being or already designed). An ADD is defined by four spaces:

- *the task space* represented by the interaction *descriptions* (IDs) that provide the way the product should be used, a procedure to follow for example;
- *the activity space* represented by the interface objects (IOs), connected to IDs that enable the user to interact with the product; for example a pilot can experience a software prototype of a cockpit instrument;
- *the evaluation space* represented by the contextual links (CLs) between IDs and IOs, these links provide the possibility of annotation;
- *the rationalization space* represented by the design rationale (DRs), as well as choice criteria and possible alternatives.

Interaction descriptions (IDs) of an ADD represent the basic specification of the user-product dialogue. IDs can be expressed either in natural language, or in a domain-specific technical language going from textual descriptions in simplified English (operational procedures for example) to a knowledge representation such as interaction blocks for example [6]. Among other things, using interaction blocks enables semi-formal testing of interaction complexity.
Interface objects (IOs) of an ADD provide an appropriate, useful and natural illusion of the product. IOs provide the ADD with capabilities of interaction and simulation enabling concrete visualization of dynamic aspects of knowledge such as color changes related to a specific semantics or evolutions of continuous parameters. IOs enable users to test product usefulness and usability by following interaction descriptions. For example, if the use of a product requires too much learning or is seldom interesting, it may not be used at all. IOs may be the components of an intermediate prototype or the final product.

Either the user follows IDs and produces an activity by using related IOs; either the user interacts directly with IOs and verifies the validity of related IDs. In both cases, usefulness and usability evaluation of IOs and IDs can be carried out and stored in *contextual links* (CLs). Evaluation can be done either in free text, or by filling in forms expressing usefulness and usability criteria. In the second case, evaluation can be quantitative or qualitative. The evaluation space enables the evaluators to annotate judgments and viewpoints on the product being developed.

The rationalization space includes design rationale descriptions, i.e., why IOs and IDs have been generated and implemented in the current ADD. The rationalization space can be more or less formalized going from textual descriptions to semi-formal representations similar to gIBIS (*graphical Issue-Based Information System*) [15] [9] or QOC (*Questions Options Criteria*) [11].

ADDs are not only communication and mediation means, but also support to prototyping and evaluation. Design knowledge is stored according to a concrete (positivist) formalism enabling the organization to stay aware of its own processes and products, based on experience. The perspective of using ADDs in an Intranet such as a design support leads to organizational issues.

K-CAP'05, October 2-5, 2005, Banff, Canada.

Even if the use of electronic documentation is recent in industry, its generation and its solution remain close to usual practice for paper documentation. ADDs serve as support to the visualization of the current state of development, user training, and knowledge reuse. ADDs have already proved their utility [5].

ADD management systems

The use of ADDs brings two important aspects to classical practice of industrial prototyping and more generally to collaborative work toward product maturity:

- interactivity and fast feed-back from user to designer; indeed, classical prototyping does not support rapid communication between designers and test-users; the availability of ADDs on an Intranet supports such communication and reactivity;
- an active space-time relation between actors along with the life cycle of a product; indeed, an ADD is living entity that enables simulation, annotation, modification and traceability.

These two aspects are taken into account in the following ADD management systems: *ADD-GEM* and *ADD-TRAC*.

PARTICIPATORY DESIGN: ADD-GEM

ADD generation and maintenance enable domain actors to share concepts by "writing" and "reading" them (in a multimedia sense), and to be in charge in the product design/use/evaluation spiral. This approach is in line with Muller's arguments that supports participatory design [13]:

- combine various sources of expertise;
- formalize personal inputs and the engagement of each actor of the life cycle of the product;
- have people who will be effectively affected by the product participate in decision-making.

The main difference between classical human-factors-driven design and *participatory design* is that in addition to the current-activity analysis, life-cycle actors self-train through ADD-supported cooperative work, for example. The former is based on observation; the latter is based on interaction and cooperation. The joint implementation of ADDs and the GEM (*Group Elicitation Method*) [3] gave birth to the ADD-GEM system that supports participatory design.

GEM consists in gathering opinions from a set of actors of the life cycle of the product being developed both by writing (*brainwriting*) and orally by structuring viewpoints into concepts. These concepts are then ordered. GEM leads to consensus and divergences elicitation within the group of actors. ADD-GEM supports a group of actors connected among each other through a computer network. Even if it may be used synchronously or asynchronously, in the same place or in different places, GEM is commonly used synchronously in the same place. In the beginning of a GEM session, an ADD is presented to each participant for usefulness, usability and acceptability testing (i.e., product maturity testing). After a first evaluation, the annotated ADD is sent to another participant according to a strategy that could be random or preset. At this point, each participant is faced with a set of viewpoints on the artifact being evaluated. He or she may validate or contradict these viewpoints or generate new ones.

The first phase (*brainwriting*) ends when each participant receives the ADD he or she evaluated in the first place. A facilitator conducts the structuring phase that transforms viewpoints into consensual concepts. He or she requests from the participants to read each generated viewpoints in a preset order. An oral discussion starts to state on the validity of a generated concept. Terminological modifications may be made. A concept can be simply rejected (this does not happen frequently in practice). Once the participants validate a concept, the facilitator requests the reading of another viewpoint and so on. The structuring phase ends when all viewpoints have been discussed and transformed into concepts. The role of the facilitator is obviously crucial to regulate the discussions and have the work done on time (usually two hours for the structuring phase). The facilitator is a conductor in the musical sense as we already described the metaphor in this paper. Concepts are typically shared on a shared projected space in real-time in order to increase cooperation. It is always possible to backtrack on previously generated concepts [3]. The next phase consists in individual concepts scoring. Individual scores are collected and processed to deduce a consensus with respect to relevant criteria. After the debriefing of the ordered concepts, the final result usually consists in a consensual ontology of the domain being discussed during the session.

The synchronous version in the same place of ADD-GEM is an electronic version of the classical paper-based GEM technique. The three other versions are really innovative and empower the ADD approach. Aspects of interactivity and active memory are predominant in them. The synchronous version in different places does not differ much from the previous in the brainwriting phase. However, since the participants are not face-to-face, the structuring discussion that contributes to the transformation of viewpoints into concepts may be very "virtual" as a group discussion by email would be.

Location does not distinguish the asynchronous version of ADD-GEM. The asynchronous version introduces delays in participant feedback. When design is bounded in time, deadlines should be given to participants for their feedback. This version of ADD-GEM provides more flexibility to participants since they may work on the topic when they are available and even in their office. However, handing deadlines is crucial issue and should be learned and mastered.

It is too soon to conclude on the type of task that should be promoted by each version of ADD-GEM. However, it is

clear that important decision tasks that validate a design solution are facilitated by the use of the synchronous version in the same place. The asynchronous version can be used in preliminary design steps or in the refinement of a solution. The same applies for the synchronous version in the different places.

TRACEABILITY: ADD-TRAC

ADDs are interconnected through a traceability mechanism that constitutes the kerned of ADD-TRAC system. ADD-TRAC uses the four spaces of an ADD, i.e.,

- the task space;
- the activity space;
- the evaluation space;
- the rationalization space.

When the user selects the traceability mode, ADD-TRAC is activated and usable. In this mode, the main entities of an ADD are interpreted by ADD-TRAC as entities to be traced, and no longer entities to be performed. In other words, when the user selects an entity, he or she is able to visualize and understand its evolution from its creation. It is always possible to return in the performance mode at any time to manipulate the selected entity in order to better understand its usefulness, usability and acceptability. It is possible to add complementary annotations in the contextual links when the evaluation mode is selected.

Task space to activity space traceability

Traceability from the task space to the activity space enables one to follow the evolution of design and evaluation of interface objects of the product (following the operational procedures associated to the product). It is possible to follow the joint design history of artifacts and procedures from simple mouse clicks.

Design decision history

There are five ways of making explicit the design decision history from the rationalization space:

- *Hierarchical indexing* based on the ADD name and the metaphor of the table of contents of a book. For example, this type of access works well when the user is a member of the design team and is familiar with the ADDs being developed.
- *Alphabetical indexing* based on a keyword list and the metaphor of the index of a book or an encyclopedia. In this case, an ADD user may select a keyword to obtain a set of ADDs that include this keyword.
- *Historical indexing* based on the creation date of the ADD or on the evaluation test period and exploited through a GANT diagram that presents ADDs generated during the project. For example, the GANT diagram can be used to focalize on a probable period of modification of the prototype as a function of an estimated period of usability tests.
- *Logical indexing* based on the design rationale text of the ADD. For example, a user may select a word or a group of words in the field of the design rationale of the ADD and look for the ADDs that share it.
- *Relational indexing* based on the direct hypertext links from an ADD to other ADDs. Such links are generally generated during the consultation of an ADD. A direct hypertext link includes the name of the person who generated it, the creation date, the creation rationale, its origin and destinations.

Traceability from the evaluation space

The evaluation space includes the viewpoints generated during the evaluation sessions. These viewpoints may be expressed formally (e.g., according to the satisfaction of product maturity criteria) or in natural language. In addition to a textual indexing (e.g., using keywords), it is possible to trace the ADDs according to maturity criteria. This possibility provides the user with exploration capabilities of all ADDs that were tested with respect to these criteria.

We have seen examples of possibilities of tracing design decisions. At any time, it is possible to contextualize an ADD, i.e., link an ADD to other ADDs or generate annotations, during a traceability investigation. The availability of traceability links between ADDs and their contextualization is essential to improve what has been done and what is going on in the life cycle of the product. Some traceability links are useful because they are frequently used and should be privileged, even if the user is always free to explore other links at any time. Other less-used links may not be presented to the user in fast-traceability mode. This feature was developed in the CID system [2].

Each ADD is considered as a referent. Each keyword or clickable part of an ADD is considered as a descriptor of this ADD. It is then important to declare these descriptors. Such declaration can be done automatically in some cases. Such a process of descriptor declaration has been presented elsewhere [2]. When the descriptor-referent relations are defined, they become usable. They are implemented using the interaction block representation, i.e., each descriptor-referent relation can be reconfigured with respect to the context of use. In other words, the underlying hypertext system articulate, accumulate and appropriate context, in Tuomi's sense [16]. It results that its user is able to anticipate some situations and act consequently. The system learns from failures and successes of users. Its maturity depends on the relevance of the contextualized descriptor-referent relations. It is only from their use that their relevance can be tested, and that the system can reproduce itself in the autopoietic sense.

For example, if a user tries to understand why an interface object has been designed the way it is and not another way, he or she will select this object, in the traceability mode.

ADD-TRAC will then present the GANT diagram of the history of the relevant ADDs. For example, the content of an ADD may be useful with respect to a specific evaluation criterion because it offers a set of modification reasons related to this criterion. In other cases, relations between ADDs are essential to understand the genesis of an interface object. When someone already made this search, it is possible to keep it as a contextualization of the traceability links used for it. In an analog context, another person can then benefit from some results of a previous search.

DISCUSSION

Contextualizing knowledge in participatory design is a crucial issue in industry to improve the efficiency of traceability. Current trend is to focus on the reengineering of industrial processes. These processes produce and use documents. These documents are generated by and targeted to those (technical and administrative) who have different viewpoints and interests. Documents are produced in five main sectors of the life cycle of a product: design, production, legal issues management, sales and support services. Support usually includes training, operations and maintenance. Legal issues include certification, incident and accident investigations.

A *chunk* of knowledge can be characterized by its content and its context of use. We saw how ADDs enable the capture of content. As content is established in a design context that is different from use context, ADDs provide the ability to store both design context (rationalization space) and use context (evaluation space). In addition, they enable the contextualization of their own use (contextualization of hypertext links between ADD versions). This accumulation of contexts of different natures forces the elicitation of relevant contexts in order to shape, format, access, understand and use contents. Two types of context elicitation may be distinguished:

1. *Different uses of the same content*. Generated annotations provide use contexts of content with associated success and failures, as well as suggestions of modifications. In this case, contextualizing is explaining a use rationale with respect to specific situations. Indeed, it is possible to incrementally generalize some parts of use rationale.
2. *The joint use of several contents*. Some contents may have been generated independently. In some cases, they constitute essential sources of information for the creation of a new content. Original contents may be directly concatenated or transformed before integration. In all these cases, original contents, transformed or not, may be associated to the new content to assure design traceability.

It is clear that hypertext technology enables the improvement of the practice of annotations (context elicitation --type 1) and links between produced documents in each sector (context elicitation --type 2). Hypertext enables the generation of new functions enabling traceability of distributed information within a network. In industry, this network is composed of different sectors producing their own traceability links. A study was conducted at EURISCO that enabled the categorization of traceability links according to three domains [19]:

- *requirement engineering* that is focused on issues of creation, formalization, refinement, storage and search for design requirements and specifications;
- *quality assurance* that focused on production control according to design requirements and specifications, as well as the documentation of the production process and the documentation of the product itself;
- *experience feedback* that focuses on the gathering of appropriated data from users operating the product, to feed other actors of the life cycle of the product and produce organizational rules. These actors are human operators, training or maintenance personnel.

Organizational intelligence and for that matter maturity often reflect the level of categorization of actors knowledge and their relations. In most organizations, actors may be human beings or machines. ADDs may be software agents or vectors of relations among human agents.

The implicit culture of an organization constitutes by itself an important part of use context of useful knowledge of this organization. This organizational culture corresponds to an implicit human memory, instead written rules correspond to an explicit human memory. It is characterized by usages that are procedural knowledge, often not formalized, defining the way other types of rule-based knowledge should be used. Contextualization of links between ADDs constitutes a support to (and an augmentation of) this implicit culture. This culture is a question of interaction style that often concretizes in the form of mediation materials between actors of the life cycle of a product. ADDs provide the necessary properties enabling the capture of this culture in the form of implicit and explicit links between interaction descriptions, interface objects, evaluations and design rationale. Any ADD has direct access to the culture of the organization thanks to interactivity with the history of the life cycle of its products.

ACKNOWLEDGMENTS

Many thanks to Helen Wilson for her help in reading this paper and her insightful comments.

REFERENCES

[1] Armengaud F., *La Pragmatique,* Que sais-je?. Presses Universitaire de France. Paris. 1993.

[2] Boy, G.A., Indexing Hypertext Documents in Context. Proceedings *of the Hypertext'91 Conference*, San Antonio, Texas, December, 1991.

[3] Boy, G.A., The Group Elicitation Method, *Proceeding of EKAW'96*, Springer Verlag, Berlin, 1996.

[4] Boy, G.A., Documenting the design process and the solution, Group Report, *ACM-DIS'97 Conference*, Amsterdam, The Netherlands, 1997.

[5] Boy, G.A., Active Design Documents. *Proceedings of DIS'97 (IHM'97),* Amsterdam, The Netherlands, ACM Press, New York.

[6] Boy, G.A., *Cognitive function analysis*. Ablex, Distributed by Greenwood, CT, 1998.

[7] Boy, G.A. Theories of Human Cognition: To Better Understand the Co-Adaptation of People and Technology - Knowledge Management, Organizational Intelligence and Learning, and Complexity, in L. Douglas Kiel (Ed.), *Encyclopedia of Life Support Systems (EOLSS),* Developed under the Auspices of the UNESCO, Eolss Publishers, Oxford ,UK, [http://www.eolss.net], 2002.

[8] Bradshaw, J., *Software agents*. AAAI/MIT Press, Cambridge, MA, 1997.

[9] Conklin, J. & Begeman, M.L., gIBIS: a hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6, pp. 303-331, 1988.

[10] Levinson S.C., *Pragmatics,* Press Syndicate of the University of Cambridge, Cambridge, UK. 1983.

[11] MacLean, A., Young, R.M., Bellotti, V. & Moran, T., Questions, options and criteria: Elements of design space space analysis. *International Journal of Human-Computer Interaction*, 6, pp. 201-250, 1988.

[12] Maturana, H.R. & Varela, F.J., *Autopoiesis and cognition: The realization of the living*. Dordrecht: Riedel, 1980.

[13] Muller, M., Participatory design in Britain and North America: Responding to the « Scandinavian Challenge ». In *Reading Through Technology, CHI'91 Conference Proceedings*. S.P. Robertson, G.M. Ohlson and J.S. Ohlson Eds. ACM, pp. 389-392, 1991.

[14] Norman, D.A. *The invisible computer*. MIT Press, Cambridge, MA, 1998.

[15] Rittel, H.W.J., Second generation design methods. Interview in Design Method Group Anniversary Report: DMG Occasional paper. Vol. 1, pp5-10, Reprinted in N. Cross, Ed. *Developments in Design Methodology*, pp. 317-327, Chichester Wiley, 1972.

[16] Tuomi, I., *Corporate knowledge: Theory and practice in intelligent organizations*. Helsinki: Metaxis, 1999.

[17] Wiener, N., *The human use of human beings: Cybernetics and society*. New York: Avon Books, 1967.

[18] Wiener, N., *Cybernetics: Control and communication in the animal and the machine*. Cambridge, MA: The MIT Press, 1975

[19] Boy, G.A., *Traceability*. EURISCO/Airbus Industrie Technical Report no. T-99-060, EURISCO, Toulouse.