Continuous capitalisation of design knowledge

Nada Matta[1], Benoit Eynard[2], Lionel roucoules[2], Marc Lemercier[3]

[1]Tech-CICO, Université de Technologie de Troyes
12 rue Marie Curie, BP. 2060, 10010 Troyes Cedex, France
e-mail: nada.matta@utt.fr

[2]LASMIS, Université de Technologie de Troyes
12 rue Marie Curie, BP. 2060, 10010 Troyes Cedex, France
e-mail: {benoit.eynard, lionel.roucoules}@utt.fr

[3]LM2S, Université de Technologie de Troyes
12 rue Marie Curie, BP. 2060, 10010 Troyes Cedex, France
e-mail: marc.lemercier@utt.fr

ABSTRACT

Learning from past projects allows designers to avoid previous errors and to solve problems. Several methods have defined techniques to memorize lessons and experiences from projects in what we call project memory. This paper presents our traceability approach that allows to extract knowledge without perturbing designers' activities. Our approach is based on web technologies. In the one hand it keeps track of knowledge produced while using design tools (as a behavior model) , in the other hand, it restitutes knowledge according to a contextual situations recognition.

Keywords

Knowledge capitalization, design knowledge , project memory, product, process

1 Introduction

Knowledge management (KM), first considered as a scientist stake becomes more and more an industrial stake. It is a complex problem that can be tackled from several viewpoints: socio-organizational, financial and economical, technical, human and legal [Dieng et al, 2000]. It concerns theoretical and practical know-how of groups of people in an organization. KM is defined as a continuous process of knowledge explicitation and internalization [Nonaka et al, 1995].

There are two types of techniques that help to make knowledge explicit (Figure 1 .):

1) Knowledge capitalization, with which knowledge can be extracted by interviewing experts and from documents. Knowledge engineering methods are mainly used in this aim [Dieng et al, 2000].

2) Direct knowledge extraction, in which knowledge are extracted directly and dynamically from organization activity. DataMining, Textmining, tracability are some of these techniques.

For instance, some studies focus on how to keep track of an activity and especially a project. In this type of studies, the challenge is how to capitalize knowledge without perturbing actors' activities and workspace. Main questions can then arise: how to extract knowledge directly from tools and documents ? How to keep track of the issue and the evolution of a project ? How to quickly model this knowledge and represent it in a way that can be easily accessible and usable by organization actors.

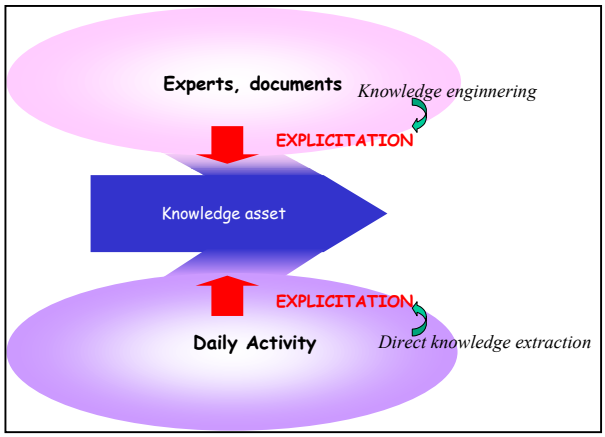

Figure 1 . Two techniques to make knowledge explicit

In this paper, we study the second type of knowledge management (direct knowledge extraction). We focus on knowledge management of a design project in order to define, what we call, design project memory (PM). A project memory can be defined as lessons and experiences from given past projects [Matta et al, 2000]. Keeping track of this knowledge can be considered as a direct extraction from several knowledge sources: documents, data bases, drawing and prototypes, meetings, activities (Figure 2 .).

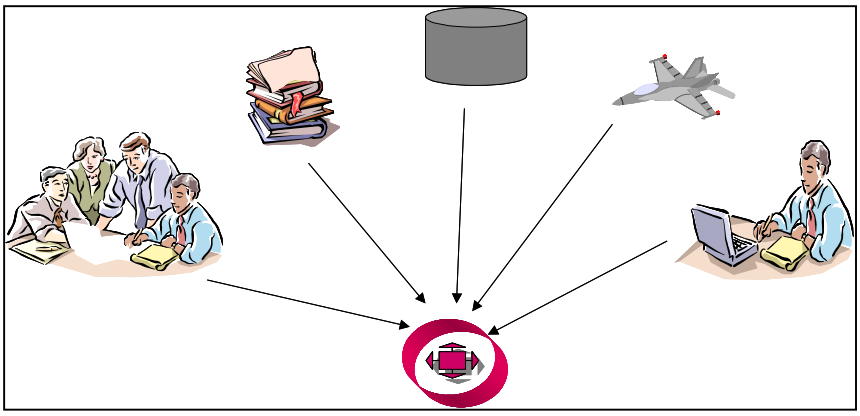

Figure 2 . Traceability of design activities

We present in this paper, traceability of engineering designer's activity. Our aim is to extract knowledge from designer's activity without perturbing him. So, we study a Web architecture that helps to define a scenario of a designer's behavior, regarding a given problem, by keeping track of used functionalities and issued information and data. Before presenting this architecture, we describe in the following section, the structure of a project memory in design.

2 Design knowlegde

2.1 Knowledge modelling in design engineering

Continuous capitalisation in engineering design consists in memorising specific information that will be later on reuse in future product designs. This information is extracted from different knowledge during design process. This dynamic knowledge of the collaborative design activity is then formalised in a static project memory (Figure 3 .). The extraction and the formalisation have to be done with a maximum of transparency for designers. Thus, they would not have to manage any extra task in the design activity.

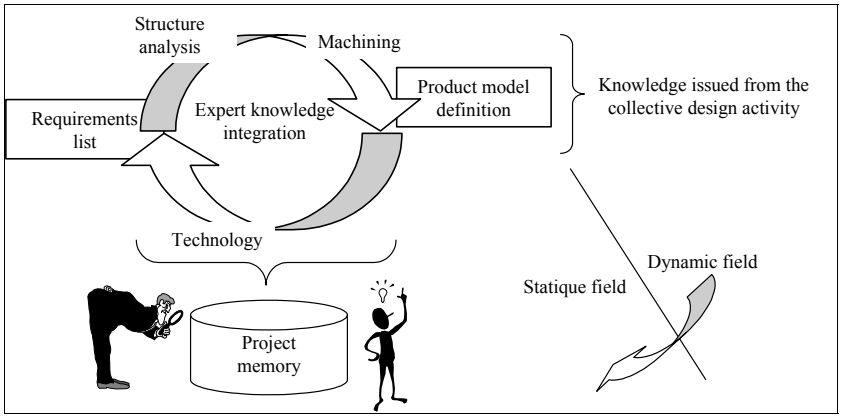

Figure 3 . Information capitalisation in engineering design.

This paper does not aim at presenting a global solution for all kinds of engineering information that must be capitalised but focuses on:

- Product data.
- Design process data.
- Design rationale data.

2.1.1 Product modelling for integrated design

Design activity is currently managed by a large group of designers that must share their points of view in order to have the product definition emerged from common decisions. Based on this Concurrent Engineering concept [Solhenius, 1992], one goal of our research works on product and process modelling is to support the progressive product definition issued from multiple points of view knowledge integration (Figure 3 .). In other words several designers have to share their knowledge (structural analysis, technological information, machining knowledge, etc.), to define and to integrate new data on the product definition. In this way, we aim to proof that the product and particularly its geometry can be totally specified by knowledge integration from the requirements list. Thus, each data is well justified and can be really taken into account in design reuse.

Design activity is a progressive mapping of product functions to product technologies. These technologies are relating to mechanical components, machining technology, etc. According to the literature, three design phases (conceptual, embodiment and detail design) have been commonly

accepted. Nevertheless, these phases are managed sequentially [Pahl et al., 1996], using axiomatic mapping [Suh, 1990] or concurrently [Andreasen et al, 1987].
Based on an integrated design method, our product modelling tries to support strong links between functions and detailed product data [Eynard, 1999] [Roucoules, 1999]. This model is quite similar to the mostly feature-based presented by [Kjelberg et al, 1992] [Krause et al, 1993] or [Anderl et al, 1996]. Indeed, feature presented as "a semantically endowed object that accompany product development from the customer request through to product release" [Shah, 1991] is very useful to define the multiple views product breakdown (cf. 2.1.2).

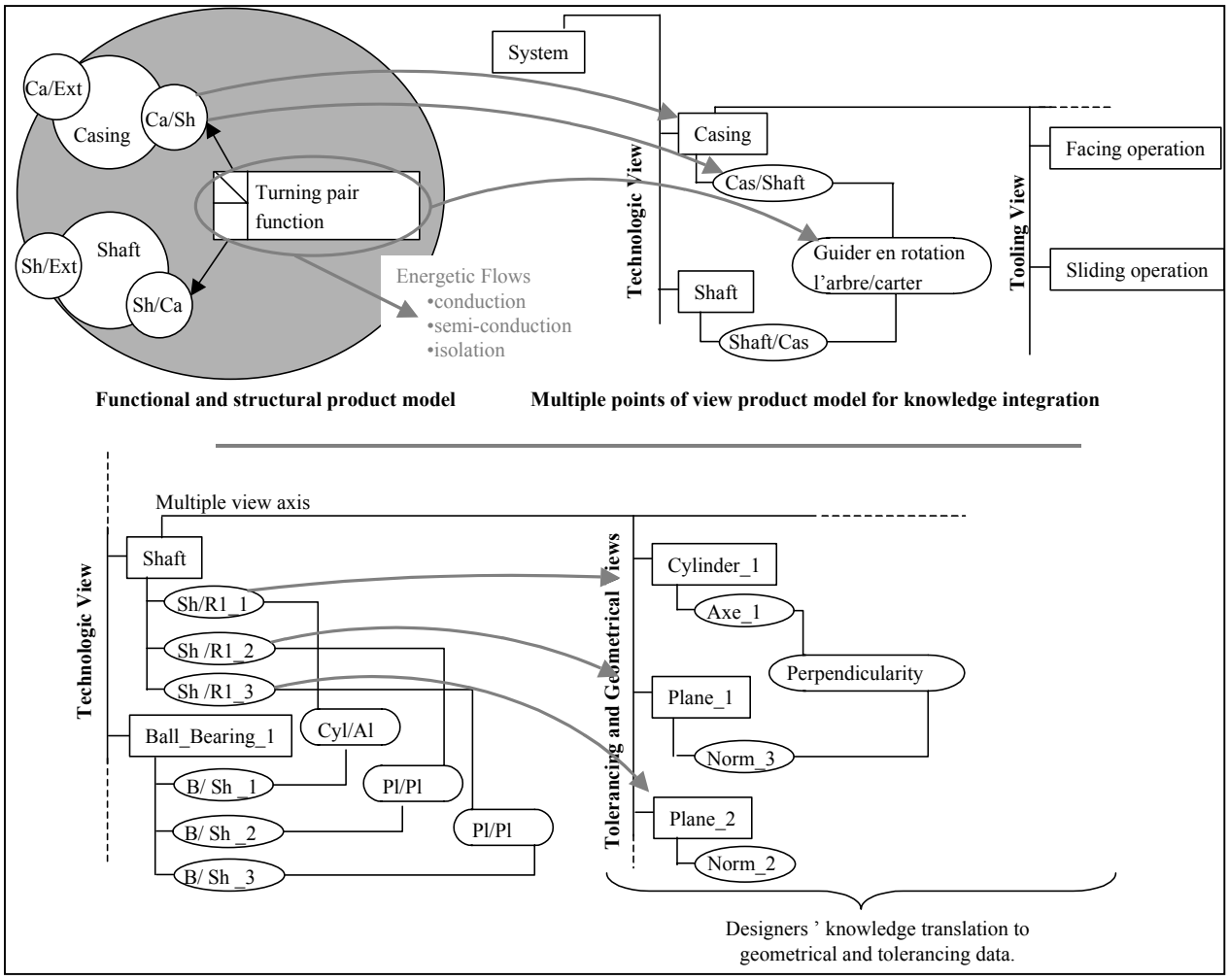

Figure 4 . Product models as data support in conceptual, embodiment and detailed design.

2.1.2 A strong link between functions and structure

For conceptual and embodiment design, a function-structure model is presented. This model is a mix of several models that describe the functional and structural representations of the product. This representation is on the one hand based on bond-graph theory to treat every kind of energetic field in the product. On the other hand the representation includes graphics and rules issued from Value Engineering tools as FAST diagram (Function Analysis System Technique). This model as presented on Figure 4 is used to progressively map product functions to product structure. Each function of the FAST diagram is linked to an energetic field that is kept coherent using the bond-graph theory.

2.1.3 A multiple points of view product definition

For embodiment and detail design a model for multiple view breakdown of the product is used. These feature-based decompositions complete the product definition adding new data and new constraints from specific points of view as Machining, Structural Analysis, etc. The model for multiple points of view is fully described in [Tichkiewitch, 1996]. As shown on figure 4, this model represents on the one hand the structural breakdown according to the function-structure product model. This view is called the *Technologic view*. On the second hand, it is easy to create and represent new views (new decompositions) of the product (e.g.: the Tooling view).
Finally, to have the product geometry emerged, the multiple product views are translated to both tolerancing and geometric views. These two common views appear then as the result of knowledge integration.

2.1.4 Computer based support for product modelling

In order to create the project memory and the continuous capitalisation (see section 3), it is necessary to manage a lot of product models. This management must also be computer-based in order to improve the transparency of the capitalisation. Therefore, extra functionality (see section 2.1.4) are added to an already-tested Co-operative Design Modeller (CoDeMo).

CoDeMo [Roucoules et al, 2000] has been developed to support the product modelling previously presented. It actually supports every product data that are managed via a server agent. Each designer can access and modify the product models via a client application. Computer developments of CoDeMo are based on C++ libraries provided by ILOG[1] Company. The functionality and features of CoDeMo (Figure 5 .) can be summarised as follow:

- To aid the creation of a product model using a Graphic User's Interface (GUI);
- To display the product data according to several representations (functional, geometrical…);
- To manage the database and propagate data constraints. Change notifications mean that each creation, deletion or modification are propagated from the server to every client;
- To support a Client/Server architecture in order to assist the co-operative work. The connections are currently done with RPC protocol but will be upgraded using CORBA technology.

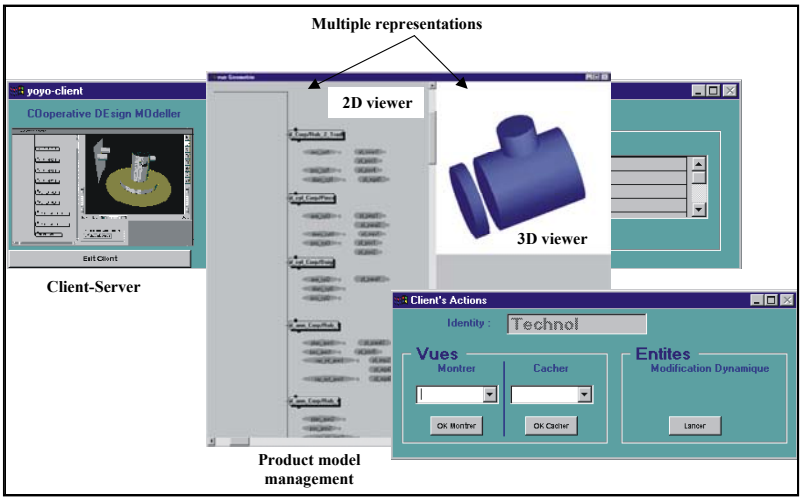

Figure 5 . Functionality of a Computer-Supported Co-operative Design Modeller.

[1] www.ilog.com.

2.1.5 Extra functionality for continuous capitalization

In the objectives of continuous capitalisation, two extra developments have been specified on CoDeMo. On the one hand (Figure 7 .), both product and process models have to be linked. This link has to be computer supported. On the other hand it would be interesting to manage product model via XML language (see section 3).

To link product and process models would be benefit in order to manage every modification applied on the product definition. This management would step by step create an history of the product model evolution during the design process. Thus it would be easier to reuse part of already created product models in future design projects where the same design problem would appear.

2.2 Modelling of Design Process

In order to have a better understanding of product development process and design activities, it is often necessary to provide details of their organisation, progress and behaviour [Eynard, 1999]. In this section, we detail briefly various modelling languages (IDEFØ, IDEF3, Petri nets, GRAI nets and UML State Diagram) before making a rapid comparison and argue of our choice for GRAI nets.

2.2.1 Process modelling language

With IDEFØ [Colquhoun et al, 1993], we get a modelling language with an efficient and simple use. It provides a good graphical representation of key elements of an activity. The activity is described with a box containing an active verb characterising the activity nature. A network of arrows links the boxes and details the relationship between activities. In this relationship, activities exchanges information or objects.

IDEF3 is the issue of a research project on information integration for concurrent engineering [Mayer et al, 1992]. The authors propose the description of process flow, precedence and causality relationship of activities and their logical junctions. The description of process flow uses the process flow network and is complemented with a representation of object state transition network. These two components allow to capture the behaviour and performance of process.

Petri nets [Murata, 1989] provide a structured description of process behaviour and allow performance assessment with associated mathematics tools. They are composed of two types of nodes: place and transition. The nodes are connected by direct arrows which specify the sequencing logic of the process. The place nodes could describe states of information or objects. The transition nodes represent operations or activities which are carried out on information or object.

GRAI nets [Pun, 1992] are based on three concepts: state or result, activity and support. States describe inputs and outputs (material or informational) of a transformation carried out by an activity. Activities represent operations performed between two successive states. Supports define all resources nature used by the activity. The graphical formalism could be translated in mathematical formalism thanks to the vectorial nature of states and supports : $\partial_i : (q_{i-1}, x_i) \rightarrow q_i$. GRAI nets provide specific models dedicated to discrete activity description, offering a satisfying characterisation of activity and having strong developments in terms of decision-making modelling.

Unified Modelling Language (UML) is a modelling language based on object oriented technology [Muller, 1997]. This language gathers the various object approaches to enable software engineering modelling. For process modelling, the UML State diagram benefits from the reference and

standardised approach of object oriented technology. It provides a state-event language and allows the modelling, analysis and specification of processes.
The GRAI nets combine the main quality of the previous modelling languages but require some developments in order to take into account all dimensions of engineering design. With the clarification of activity nature between states, the model benefits from logical link with the product modelling [Eynard, 1999]. Based on the information captured in GRAI nets, we are able to represent the behaviour knowledge and process sequencing and actions of design team, etc.

2.2.2 Modelling of key elements of design process

[Eynard, 1999]specify an extension of GRAI nets oriented to product development process modelling. He identifies three kinds of activities: design, execution and decision-making. The input and output states detail the information transformed by activities.

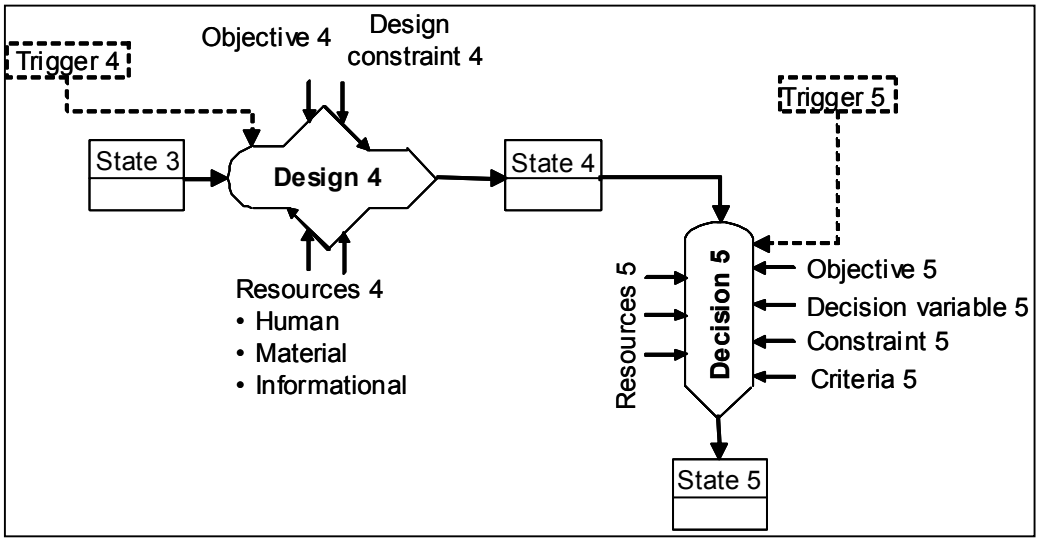

Figure 6 . Sequencing of design and decision-making activities

1- The design activity (Figure 6 .) can be defined by its iterative, creative and basically human character. It includes the understanding and analysis of problems, and the search for, creation, synthesis and proposal of solutions. The design activity is characterized by:

- the information transformed by the activity, which is represented by an input and an output state;
- the activity supports, which are of three types: material, informational and human resources;
- the specific support of the design activity, which is the design framework i.e. objectives and design constraints.

2- The execution activity is characterized by its procedural and often programmable or computational nature. It can describe the detail design of a part, the drafting of a document, etc. The execution activity is characterized by:

- the information transformed by the activity, which is represented by an input and an output state;
- the activity supports, which are of three types: material, informational and human resources.

3- As design, the decision-making activity (Figure 6 .) has a basically human character but it is purely decisional. This activity makes choices and decisions and selects alternatives in the development process. The decision-making activity is characterised by:

- the information transformed by the activity, which is represented by an input and an output state;

- the activity supports, which are of three types: material, informational and human resources;
- the specific support of the decision-making activity, which is the decision-making framework i.e. objectives, decision variables, constraints and criteria.

2.2.3 Link between product and process

Regarding the product development process, our aim is to capitalize the design history. This design history will be based on product and process modeling detailed above. It will provide a support to designers with the key elements of design project. The product dimension will be based on CoDeMo with a progressive history of product definition. The process dimension will provide a detailed description of activities, the organization and planning of the project according to [Shah et al., 1996] and [Blessing, 1996] viewpoints.

The continuous capitalization will ensure a quick and efficient knowledge capture. The capitalization of knowledge related to product will be transparently done for designer through CoDeMo. The process modeling will provide a detailed description of transformed flow, activity support, sequencing, behavior, etc. Thus based on these three dimensions of capitalization will obtain a strong environment of capture, modeling and reuse of design knowledge.

2.3 Design rationale

Design rationale can be defined as the rationale space for problem solving. This space concerns individual and collective dimensions. Generally, discussions, alternative choices, problem solving are fleeting knowledge in a project. Nowadays the challenge is to define methods and tools in order to represent the rationale of a project and to memorize it. This type of knowledge can be characterized as :

- Problem definition: subjects, type, elements.
- Problem solving: participants, methods used and potential choices.
- Solution evaluation: rejected solutions and arguments, advantages and disadvantages.
- Decision: solution and arguments, advantages and disadvantages.

Several methods have studied how to capitalize problem solving knowledge by emphasizing the problem treated, the potential solving choices and arguments. We note for example in one hand, IBIS, QOC, DRAMA that represent the design rationale as decision space and in another hand DIPA and DRCS that suggest a problem solving modeling. Reader can have more details in [Matta et al, 2000] about these methods.
In this paper, we study relations between in one hand design rationale and in another hand, product and process models. So, we do not present design rationale capture process. For more details, see [Bekhti et al, 2001].

2.4 Structure of project memory in design

A project memory in design must consider the different part, we noted above. This type of knowledge can be organized as:

- The project organization :
  - Participants, their competencies, their roles in the project and relationships
  - Process, task organizations, constraints and requirements
- The project environment:

- Project goal
- References, rules, methods and directives
- Tools and techniques

- Project realization :
  - Design rationale
  - Product description

These elements have mutual influences that is important to emphasize in a project memory (Figure 7 .).

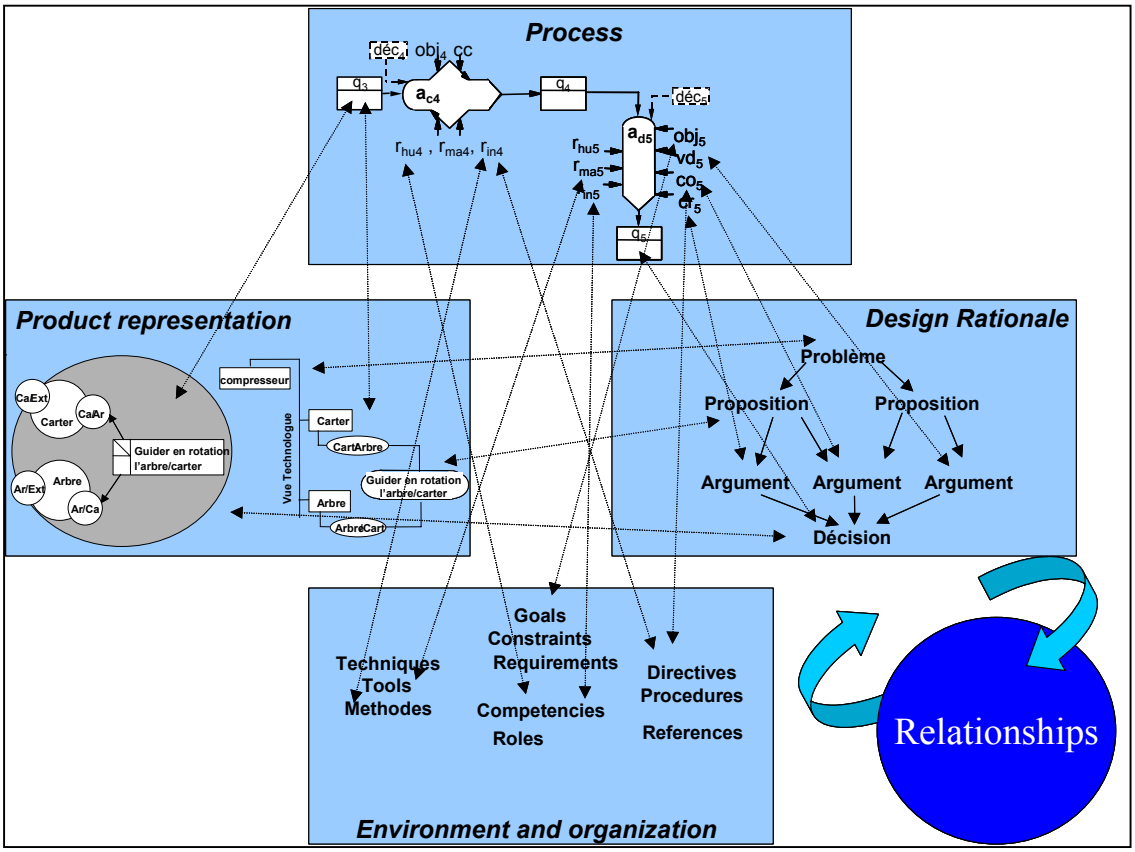

Figure 7 . Project memory structure

After presenting the different parts of a project memory, the next section describes how some of these knowledge as environment, organization, product knowledge and especially problem solving may be extracted directly from designer's activity.

3 Direct knowledge capitalisation from the activity

Currently, designers mostly work by using design software (ex: CAD/CAM), etc. , They even use innovation tools for creating new ideas (ex: TechOptimizer™). Our idea, is to extract the behaviour of designer by observing his activity when he uses software to solve a given problem. This behaviour can be kept as scenarii of used functions, corresponding data and documents produced, interactions (e-mails, data exchanges, …), etc. We specify a web architecture (described in the next section) that allows the observation of the designer activity [Eynard et al, 2001]. XML can also be

used in order to structure data extracted as a behaviour model. A knowledge engineer can then analyse behaviour models and describe environment and problem solving elements. These elements can be linked to project knowledge (constraints, requirements, organization, etc.) and enrich the project memory. Figure 8 . illustrates this process.

The observation of experts' activity and problem solving has been largely used in knowledge engineering for knowledge extraction [Aussenac, 1989]. This technique is inherited from cognitive psychology and ergonomics. In this technique, the observer needs some elements related to the global project, before starting the observation as for instance, the step of the process the expert deals with, and corresponding constraints and requirements. In order to bring out these elements, the designer is invited to identify that task he carries out when he uses software. This identification allows to establish the link between the behaviour model we observe and the project organizations and corresponding environment.

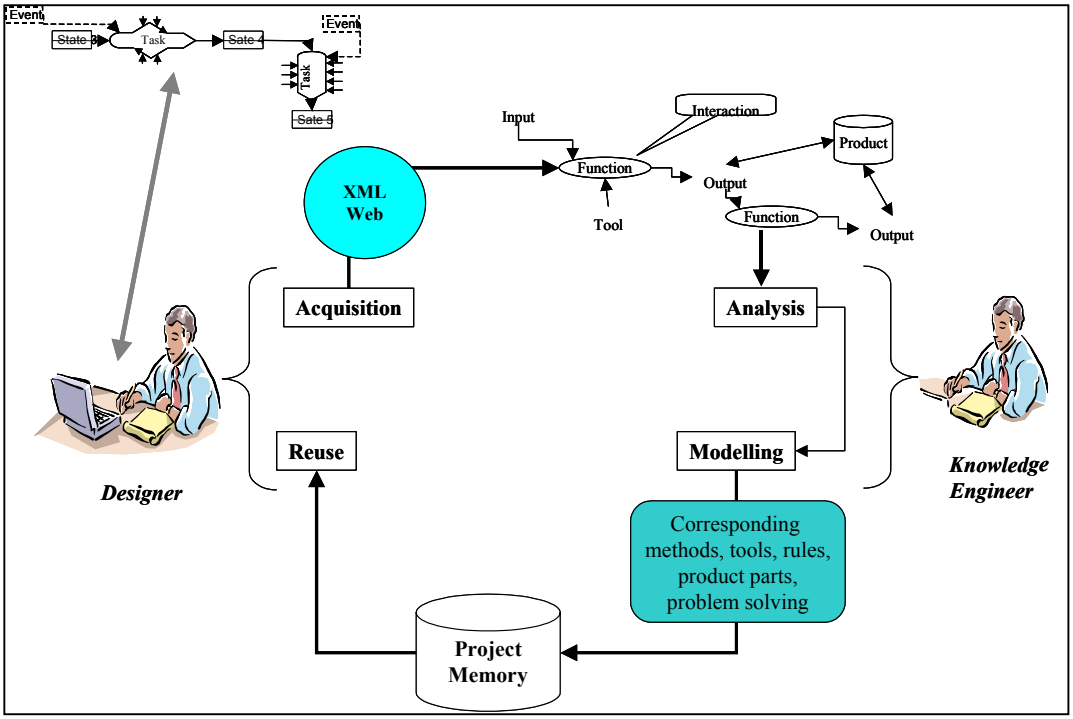

Figure 8 . Designer activity observation

We present in the following the Web architecture we defined for this aim and how it can be used not only for designer's activity observation but also for knowledge restitution.

3.1 Web architecture

In this paragraph, we present the main elements of the experimental platform developed for this project. The « *project memory* » is an application localized in one place in the set of entities participating to the project. Its role consists in recovering information linked to designers' activities. These information received are heterogeneous. We have selected the XML language as the federal language.

Our project memory software is based on both XML and Web technologies. In a first version, we have favoured the Java language because it proposes efficient solutions to insure interactions with XML and Web topics [Bernadac et al, 1999]. To manipulate directly an XML document, the SAX interface (Simple API for XML) has been required in the XML community because it proposes an

event framework. To each step of the analysis process, SAX releases an event associated to the XML element of the document. An other approach, the DOM interface (Document Object Model) has been proposed by the W3C. DOM proposes an object representation of a XML document and provides tools for the manipulation of trees. The XML document in its totality is redefined in the memory. More specifically, the JDOM API is used in the Java community. It proposes a great number of simplifications in the use of DOM by a transformation of all DOM interfaces and DOM class in real Java classes. In a Web context, the Java main proposal is the Servlet concept that has allowed the use of all Java classes in the development of complex applications linked to Web servers.

In conclusion, with the first version of our demonstrator, designers use a simple Web browser because all the software are Web applications localized on the central site (mail, agenda, document's transfer, …). For the technical point of view, this first version has been realised with an Apache Tomcat Web server and several Java Servlets [Liu et al., 2001].

The version 2 of our demonstrator is still under development. However, we have already validated several elements increasing the functionality of the first version of our demonstrator. The main limitation concerns distant applications used by designers. It is indeed probable that on each site, particular applications will be used. In this case, we have to insure the information circulation to the central site. Brought solutions depend on the applications.

3.1.1 Case 1: a Web software in a distant site

A designer uses a Web application on its site. This first case is easy to manage. We modify HTML pages by adding Javascript functions. Thus, information normally transmitted to the local Web server in the designer transferred to the project memory Web server. After information recovery, the demonstrator broadcasts these data to the first Web server.

3.1.2 Case 2: not Web open applications

In the case of software developed for our project, it is possible to add a module of data recovery. We have implemented three approaches to insure the transfer of information to the central site. The first approach consists in opening a network connection (socket TCP/IP) to our demonstrator. We have used this solution for applications enough ancient generally written in C or Pascal language. The second approach has been used for applications written in object language and especially in Java. The recovery module is a Java RMI client (Remote Method Invocation) that communicates with a RMI server localized on the central site. This RMI server is an additional element of our demonstrator. The third approach, more recent, is based on concepts of Web Services. A Web-Service is an application based on protocols of Internet that provides a specific service by respecting XML exchange format. It can also be seen as an accessible transaction by the exchange of XML documents between two sites. Web-Services represent the most promising solution for the integration of distributed services in a strongly heterogeneous context. Indeed, current solutions possess some restrictions. The DCOM solution from Microsoft imposes the choice of the Windows platform. Java RMI and Java EJB (Enterprise Java Beans) support only the Java language. Finally, CORBA, the OMG solution rests on the utilisation of interoperable ORB. The main result research with the use of Web-Services is therefore a real interoperability of all applications. Components of Web-Services [Lemercier, 1997] are mainly SOAP (Simple Object Access Protocol), WSDL (Web Description Service Language), WSFL (Web Service Flow Language) and UDDI (Universal Description, Discovery and Integration).

3.1.3 Case 3: other cases

In the case of the use of a closed software proposed by a company, the solution consists in asking an extension of this software to be able to provide information from designer's activities. This synthesis will be then translated to the XML format if that is necessary and transmitted to our project memory.

3.2 The representation of the memory using the Web architecture

The project memory can be represented as a number of XML documents. These documents can be also linked to other data bases produced by specific product design (for instance CoDeMo) and process management tools. XML documents represent in fact, a flexible indexation of these documents and an enrichment because it allows also to represent design rationale. Automatic links (XLL) can be used to establish this flexible indexation and relations between all the parts of the project memory. The style sheets XSL is a good support to present the memory in different way corresponding to the needs of the user. The representation of the project memory can be illustrated Figure 9 .

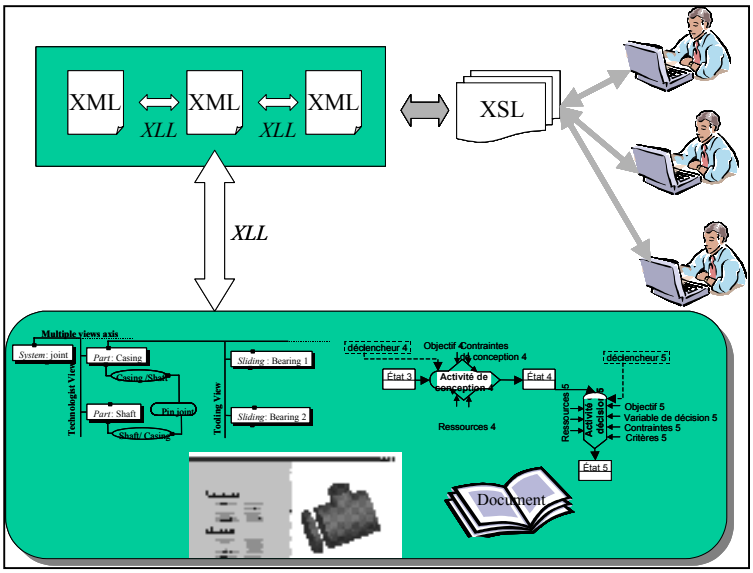

Figure 9 . A XML representation of the project memory

As we noted above, the activity observation can be also used to recognize knowledge from the memory. In fact, we plan to use a probability algorithm based on scenarii of activities in order to recognize the context of the designer and propose a contextual access to the memory and problem solving part. The project memory can be viewed as a case base in which the environment, process and product knowledge represent the case definition and design rationale represents the case solution. So, similarity research algorithm can be used for case recognition. In project memory, the similarity can be based in different elements of the context depending on the current activity. So the similarity algorithm must be flexible enough to support this type of recognition. Note also that some context elements can be included in the solution beside problem solving. We plan to test an algorithm based on the probability for this aim.

4 Conclusion

Learning from past projects allows designers to avoid previous errors and to solve problems. A number of methods defined techniques to memorize lessons and experiences from projects. We study in this paper a traceability approach that allows to extract knowledge directly from designer's activities. The basic principle of this approach is to observe a designer facing to a problem. We use web technologies in this aim, in order to establish a behavior model of the designer by extracting and linking functions and data he uses and produces. This behavior model can be then analyzed and structured in a project memory.

Our thesis is in the one hand, to keep track of knowledge without disturbing designers' activities and in the other hand, guarantee a structured and intelligent access to the memory. For that, the direct knowledge extraction as we defined, can be also used to recognize knowledge from the memory and offer a contextual restitution of knowledge. In fact, the behavior model can describe some elements of the current context and needs of the designer. Thus allow to match these elements with the memory and extract similar problem solving that can help the designer to face his problem. We plan to use similarity algorithm used in the Case Based Reasoning and Human Computer Interface techniques.

In a project memory different types of knowledge must be represented: environment description, process, product and design rationale. These elements can be structured using internal and specific representation usually adopted in engineering design. The project memory can point these elements as an intelligent index based on problem solving that is the main part of traceability. With this type of representation, we do not introduce heterogeneous representation coming primly from the cognitive and artificial intelligence science "as semantic network and cognitive models".

1. 5. REFERENCES

[Anderl et al, 1996] Anderl R., Mendgen R., "Modelling with constraints: theoretical foundation and applications", Computer Aided Design, Vol. 28, n°3, pp 155-166, 1996

[Andreasen et al, 1987] Andreasen M.M., Hein L., "Integrated product development", Springer-Verlag, London, 1987

[Aussenac, 1989] Aussenac N. – *Conception d'une méthodologie et d'un outil d'acquisition des connaissances expertes*, PhD report of the university of Paul Sabatier, Toulouse, October, 1989.

[Bekhti et al, 2001] Bekhti S., Matta N., Andéol B. et Aubertin G. – Mémoire de projet : Processus dynamique de modélisation des connaissance , *Proceedings of Cooperation, Innovation and Technologies* CITE'2001, Troyes, 29-30 November 2001, p. 329-345.

[Bernadac et al, 1999] BERNADAC J.C., KNAB F., *Construire une application XML*, Editions Eyrolles, Paris, 1999.

[Blessing, 1996] Blessing L.T.M. (1996) Design process capture and support*, 2nd Workshop on Product Structuring*, Delft, The Netherlands

[Muller, 1997], P.A. Muller, *Modélisation objet avec UML*, Edition Eyrolles, 1997.

[Colquhoun et al, 1993] G.J. Colquhoun, R.W. Baines, R. Crossley, A state of the art review of IDEFØ, *International Journal of Computer Integrated Manufacturing*, Vol. 6, n° 4, pp 252-264 (1993)

[Dieng et al, 2000] Dieng R., Corby O., Giboin A., Golebiwska J., Matta N., Ribière M., *Méthodes et outils pour la gestion des connaissances*, Dunod., 2000.

[Eynard 1999] Eynard B., "Modélisation du produit et des activités de conception. Contribution à la conduite et à la traçabilité du processus d'ingénierie", PhD thesis of the Bordeaux 1 University (France), 1999.

[Eynard et al, 2001] Eynard B., Lemercier M., Matta N, Apport des technologies internet et du langage XML dans la constitution de mémoires de projet en conception de produit, *Proceedings of Cooperation, Innovation and Technologies*, CITE'2001, November 2001.

[Kjelberg et al, 1992] Kjelberg T, Scmelkel H., "Product Modelling and Information Integrated Engineering Systems", Annals of the CIRP, vol. 41, n°1, pp 201-204, 1992

[Krause et al, 1993] Krause F.-L., Kimura F., Kjelberg T., Lu S. C.-Y., "Product modelling", Annals of the CIRP, vol. 42, n°2, pp 695-706, 1993.

[Lemercier, 1997] Lemnercier M., Développements Avancés de Pages Web Dynamiques , *NOTERE'97 (Colloque International sur les Nouvelles Technologies de la Répartition)*, p. 133-146, Pau, novembre 1997.

[Liu et al., 2001] Liu D., Xu W., A review of web-based product data management systems , *Computers in Industry*, Vol. 44, 2001, pp. 252-262.

[Matta et al, 2000] Matta, N., Ribière, M., Corby, O., Lewkowicz, M., et Zacklad, M. Project Memory in Design, *Industrial Knowledge Management - A Micro Level Approach*. SPRINGER-VERLAG : RAJKUMAR ROY, 2000

[Mayer et al, 1992] R.J. Mayer, T.P. Cullinane, P.S. DeWitte, W.B. Knappenberger, B. Perakath, M.S. Wells, IDEF3 process description capture Method, *Information Integration for Concurrent Engineering - Compendium Methods Report*, Wright-Patterson Air Force Base, Ohio, USA (1992)

[Murata, 1989] T. Murata**,** Petri nets : properties, analysis and applications, *Proceedings of the IEEE*, Vol. 77, n°4, pp. 541-580, (1989)

[Nonaka et al, 1995] I. Nonaka, H. Takeuchi: *The knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995

[Pahl et al, 1996] Pahl G., Beitz W., "Engineering design : a systematic approach", Springer-Verlag, London, 1996

[Pun, 1992] L. Pun, *Integrated discrete production control : analysis and synthesis - A view based on GRAI nets* ; Elsevier, Amsterdam (1992)

[Roucoules, 1999] Roucoules L., "Méthodes et connaissances. Contribution au développement d'un environement de conception intégrée", PhD thesis of the Institut National Polytechnique of Grenoble (France), 1999

[Roucoules et al, 2000] Roucoules L., Tichkiewitch S., "CoDE: a Co-operative Design Environment. A new generation of CAD systems", CERA journal, Vol.8, n°4, pp 263-280, December 2000

[Shah et al, 1991] Shah J., "Assessment of Feature Technology", Computer Aided Design, vol. 23, n°5, June 1991

[Shah et al, 1996] J.J. Shah, D.K. Jeon, S.D. Urban, P. Bliznakov, M.T. Rogers, "Database infrastructure for supporting engineering design histories', Computer Aided Design, Vol. 28, n° 5, pp 347-360, 1996

[Sohlenius, 1992] Sohlenius G., "Concurrent Engineering", Annals of the CIRP, vol. 41, n°2, pp 645-655, 1992

[Suh, 1990] Suh N.P., "The principles of design", Oxford University Press, New York, 1990

[Tichkiewitch 1996] Tichkiewitch S., "Specification on integrated design methodology using a multi-view product model", ESDA Proceedings of the 1996 ASME System Design and Analysis Conference, PD-Vol. 80, 1996