7

USING CONTEXTS IN MANAGING PRODUCT KNOWLEDGE

John J. Mills[1], J.B.M. Goossenaerts[2]
[1]*Department of Mechanical and Aerospace Engineering, The University of Texas at Arlington, Arlington, TX, USA; Em: jmills@mae.uta.edu*
[2]*Department of Technology Management, Eindhoven University of Technology, the Netherlands; Em: J.B.M.Goossenaerts@tm.tue.nl*

Abstract: The appropriate use of context is a powerful tool for managing complexity asexemplified by the realization of a modern product in a globally distributed environment. We show that computational contexts can be created from the properties of entities such as the company, the project, the user, his organizational role, and the product itself. Our paper briefly presents the architecture of a product knowledge environment that is based on computational contexts. It then discusses several possible uses of computational contexts in the product realization process including management of product knowledge, managing the product realization project and supporting users while they perform the wide variety of tasks necessary to design, develop and produce a modern product.

Key words: Product Knowledge, Context

1. INTRODUCTION

Several authors have pointed out that industry has to move away from thinking about managing product data to managing product knowledge. There are many viewpoints as what product knowledge and indeed what knowledge is. We have proposed elsewhere that knowledge comes into existence when someone identifies in sets of data, in information or in existing codified knowledge, one or more patterns or relationships which exist across multiple contexts. A pattern or relationship which exists only in a single context is only information. Further, we pragmatically define codified *Product Knowledge* as the combination of the set of all *data sets* which depict everything we need to know about a product, the context independent *relationships* within and among these data sets, and the *contexts* in which they are created and used. A data set is just a set of data which, when properly interpreted, depicts some aspect of the product (e.g. requirements, geometry, production plans, inspection programs). A data set can be just a file. The word "codified" is placed in front of Product Knowledge to indicate that in this work we do not attempt to deal with tacit knowledge which is an other, separate and equally important area of research.

Our approach is based on the idea of a *computational context* which seeks to mimic how humans use context. We show that computational contexts can be created from the properties of entities such as the company, the project, the user, his organizational role, and the product itself. The architecture of a product knowledge system is proposed. It is partially validated by addressing the following three issues:

- How to detect context changes?
- What areas of the process can the context help support?
- What advantages does using context have over other approaches?

2. DEFINITIONS

2.1 Knowledge and Product Knowledge

Our view of product knowledge and indeed knowledge itself is based on the concept that the progression from data to information to knowledge is through the recognition and understanding of patterns[1]. Patterns in data make information. Information becomes knowledge when some pattern is recognized and understood in someone's mind in information in more than one context. Thus, Newton created new knowledge (his laws of motion) when he observed patterns in the behavior of different objects in different contexts.

Applying these definitions to product data, patterns in data imply relationships within and among sets of data. A CAD file is just data until it is associated in some way with a pattern that brings meaning to it. STEP recognizes this with the idea of a data model which describes the specific pattern (i.e. the relationships within) the data within the file. Thus files of data in general (e.g. CAD, Spreadsheets, data bases, project plans) have no meaning until other information is associated with it: for instance, until it is associated with a data model. Product Data Management Systems (PDM's) manage the complex relationships among sets of data describing all parts of the product. Workflow management systems (WFM's) manage the complex relationships that occur in the flow of product data along the product realization process and that occur among the various tasks that are required in that process at a higher level than that of a CAD system. However, it is not clear what can be called knowledge and what information because contexts are not identified. Moreover, the approach taken in both appears to be ad-hoc with no fundamental basis. Both of these management systems manage what we call "data sets". A data set can be a CAD file, a set of requirements, an NC program, etc. A data set has no meaning until other information is associated with it. The source of this other information is discussed below after we have introduced the concept of "context".

We can identify some product knowledge immediately. One fact that is true, independent of context, is the suggestion by various authors [2] and generalized by us that a product consists of a set of "product units" arranged in some topology. This is a pattern that exists for all kinds of products. Elsewhere we have suggested that each "product unit" can be represented in a computer (or on paper or some other medium) by abstract entity which we call its "Product Unit Representation" (PUR). The PUR consists of two kinds of "Aspects"[3]: a single "Core Aspect" which contains properties which are intrinsic to that product unit and a set of Aspects which contain all the context dependent information about the product unit. The Core Aspect contains several facts that are true about the product unit, independent of context. A product unit has a color, a weight, a material it is made of and a cost. The value of all of these is independent of context because they are intrinsic to that product or component. An Aspect consists of a data set with an associated list of context dependent properties assigned to that Data set at the time of its creation or modification. A product unit has many Aspects.

2.2 Contexts

PDM's and WFM's essentially seek to manage the complex relationships among data sets involved in product realization *across multiple contexts*. Users sitting in their offices perform tasks that eventually and in totality realize a product. Each works in their own "context": a situation which constrains how they perform that task, indeed, what task they perform.

Contexts are used daily by everybody as a method of managing the complexity of their daily lives. We think nothing of switching from being father to husband to driver to navigator to executive to negotiator. Yet contexts have been woefully neglected in research until recently. Recent workshops have started to reverse this trend, and research results are becoming available in a wide variety of fields. Despite this research, there is still no universally accepted definition of a "context"[4]. Brézillon and Pomerol have recently reviewed the state of the art in context research[5]. In product realization, context has been mentioned a number of times [3,6,7].

Our current view of context is based partially on observations of how humans use contexts, albeit in an instinctive way. Humans seem to be able to build complex contexts automatically and instinctively without really knowing what they do. First the context (a pattern) is recognized and understood. Then the set or properties required to deal with that context are automatically assembled. Humans create the context from entities that surround them, extracting from those entities, properties that are relevant to themselves. The properties of the entities involved in building the context are learned by experience and constitute knowledge when those patterns of entity combinations are used in multiple context. Properties of entities can be other entities, procedures, attributes, rules, etc.. We use the word "entity" in its fundamental sense[8].

3. CONTEXTS IN A PRODUCT KNOWLEDGE SYSTEM

3.1 Computational Context

In a computational environment supporting humans performing tasks, we suggest the concept of a computational context defined as follows: *a computational context surrounding an entity of interest is a set of properties (with values), that are (a) provided by a set of entities in the same symbolic or physical space as the entity of interest, (b) relevant to the entity of interest in that situation of interest during some time interval and (c) added to the properties of that entity only within that context* [5].

The entity of interest is the thing which is surrounded by the context. It may be a person, a task, a file or a set of data. In product realization, there are two generic entities of interest: The set of data which describes some aspect of the product and the human performing the task. Computational contexts surrounding a set of data has been discussed elsewhere [9].

The set of possible entities that can form contexts for data set creation is quite large, including : the company, the operating system of the computer of the user, the industry sector to which the company belongs, the person performing the task, the role they are playing in the project, the product, the application used to create the data set, and so on. Each of these brings different properties that are relevant to the data set. The product may define the process to be used (e.g. novel or parametric).

The properties of entities forming the context can by anything including other entities, attributes, rule sets, constraints. An example is a project context which has properties such as ID, Name, Description, Responsible person, Goals and other entities such as the company, which has as one of its properties a country context. This nesting of context properties is important because it allows complex relationships to be established among entities of interest. The idea is that entities only become contexts when they are relevant to a particular entity of interest.

3.2 System Architecture

The system we are proposing to create computational contexts for people performing tasks and the aspects they create and use is illustrated in Figure 1. The Entity Data Base (EDB) contains specific entities with particular values for their properties. The Context Ontology (CO) contains rules for organizing and combining the entities appropriate to the specific context. The rules include a set which use the idea of context levels (syntax, semantics and pragmatics) as discussed in an earlier paper. The CO contains the rules which allow the PKE to govern when general entities such as user, project, company etc., become relevant to a particular entity of interest and allow the PKE to build a specific Computational Context for it. The Aspect Repository (AR) contains this specific computational context information for that particulare entity of interest, say a data set like a CAD model. The name of its file is combined with the computation context information to form the Aspect for that CAD model and this information is then stored in the Ar for future reference. The AR stores no data sets. These can be stored anywhere on the WEB. The computation context information contains the name of the file and its location so that it may readily be accessed at a later date. For enetities of ineterst such as users, the computational context would contain such information as appropriate processes for the product realization, tool lists, other similar products to facilitate their tasks. The Product Knowledge Environment (PKE) provides functions to users. This includes the ability to (a) create user computational contexts with suggested Aspects, procedures, protocols, rules, standards etc., (b) create computation contexts for data sets which then become Aspects stored in the AR and (c) allow users to search for specific data sets (i.e. Aspects) using the properties of the Aspects as indices. This latter allows users to create multiple directory structures for data sets based on the values of the properties in the Aspect. For instance the Aspects can be organized by the PKE by product structure, organization unit, project organization by team structure, all of which are properties of the computation context formed from the entities in the EDB.

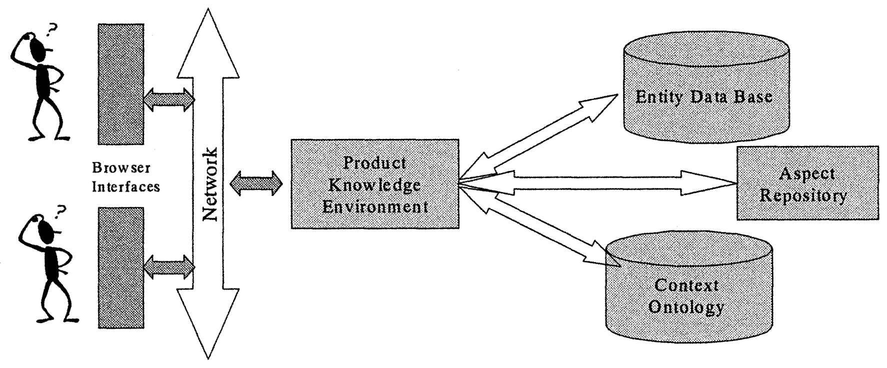

Figure 1. Architecture of a Product Knowledge Environment

3.3 Development

In order to partially develop the architecture we are addressing the following several issues:

- What advantages does using context have over other approaches?
- What areas of the process can the context help support?
- How detect context changes?
- What are the critical rules for the CO?

The main advantage of the context-based approach to supporting the product realization process is that it builts on what people to every minute of every day: detecting and using contexts to manage the complexity of their lives.

For the second issue, we believe that context-based system described above can help manage the data, information and knowledge more efficiently since the aspect concept described elsewhere allows one to organize data sets along multiple dimensions: process, project, company, industry, product structure, and can also be used to automatically provide required data sets for a particular task. With an appropriate set of relevant properties a context can supply a user with (a) sub processes and procedures that change with the users experience, training and task, (b) appropriate Aspects for a task, (c) standards that change with the industry segment and country, and (d) policies and procedures from the company and project. While not all of these will be needed for a particular task, the PKE can provide easy access to them.

Our current thinking about detecting contexts is that it will depend on the Entity of Interest. For a data set and its accompanying Aspect, we envisage a set of key properties that will be compared: The context into which it is being brought vs the context in which it was created. We would start the comparison at the most deeply nested entity and move outwards. Since most context changes for Aspects will be in the syntactic level, this will allow for fast detection, while still allowing an Aspect to be moved globally.

When a user changes context usually this will be some action taken which can be detected through keystrokes or mouse clicks. One example would be a user changing tasks by selecting a new task. Another might be when a user has multiple tasks running in different windows. The move to a different window would indicate a change in context.

4. DISCUSSION

We have briefly presented our idea on knowledge, product knowledge context and a suggested system which makes use of these ideas. As an example of how this system architecture might be used consider a CAD model. This entity of interest is created in a context created from a particular combination of particular entities such as the user, his computer environment including the application they use to create it, their task, the project phase, the project, the company, and the country in which they work. The PKE has identified this specific computation context and created in according to the rules it finds in the Context Ontology. When the user saves the CAD model, under a specific file name in a specific directory on a particular computer, the PKE collects that information as part of the syntactic context level and combines it with the rest of the computational context information (the semantic and pragmatic levels) and stores it as an Aspect in the Aspect Repository. Note that the file - a data set - can be stored anywhere on the World Wide Web. Part of the information in the Aspect is the list of tasks that need this file and the people responsible for these tasks. The PKE would then inform these people that the file is available. In this example, the PKE acts like a Work Flow Manager as well as a Product Data Manager but instead of being based on ad-hoc assignments of files to cabinets and people, it is based on natural contexts. When one of these people enters the PKE and chooses one of these tasks, the PKE knows from the Aspect where it is, how it is stored and in what format, and can immediately transfer the file to that computer and make it available to the user. It is foreseeable that the

PKE could also provide automatic translation of the file format to that of the application the new user is running.

This system has the potential for being the next generation of product realization knowledge managers. Our vision is that entities in the EDB can include high level tasks which specific procedures for dealing with certain types of product units (novel units, redesigned units), experience level of users (novice, 40 years experience) which are used by the PKE and the CO to create specific computational contexts with suggested approaches, standards, protocols, and Aspects appropriate to the users training and background. Space precludes mentioning all the possible uses that the proposed system may have in managing product realization knowledge.

ACKNOWLEDGEMENTS

J. J. Mills would like to acknowledge the financial assistance and support that he obtained from the Information Technology Group of the Technology Management Faculty of Eindhoven University of Technology and the Institute of Mechanical systems of the Swiss Federal Institute of Technology, Zurich, during the Sabbatical leave he spend with those institutions. During this time many of the ideas in this paper were created and clarified during informative and insightful discussions with their faculty and students.

REFERENCES

1. Mills, J.J. and J.B.M. Goossenaerts (2001) Towards Information and Knowledge in Product Realization Infrastructures. In: J.P.T. Mo and L. Nemes (eds.) (2001) *Global Engineering, Manufacturing and Enterprise Networks*, Kluwer Academic Publishers, Boston pp 245-254
2. Henderson, M. R. and L. E. Taylor (1993) A Meta-Model for Mechanical Products Based upon the Mechanical Design Process, Res. in Eng. Design, Vol. 5, pp 140-160
3. Talukdar, S. N. and S.J. Fenves (1991) Towards a Framework for Concurrent Design. In: D. Sriram, R. Logcher, S. Fukuda (eds.) *Computer-Aided Cooperative Product Development, MIT-JSME Workshop*, MIT, Cambridge, USA, November 20/21, 1989, Proceedings. LNCS 492 Springer-Verlag
4. Bouquet, P. L. Serani and P. Brézillon (1999) Preface to *2nd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'99)*, LNAI 1688, Springer.
5. Brézillon, P. and J.-Ch. Pomerol (1996) Context in Artificial Intelligence: II. Key elements of contexts. Available at URL: http://citeseer.nj.nec.com/502533.html
6. Kannapan, S. and K.M. Marshek (1992) Engineering Design Methodologies: A New Perspective. In A.N. Kusiak (ed.) *Intelligent Design and Manufacturing*, John-Wiley & Sons, pp. 3-38.

7. Hale, M. A. (2000) Presentation at the *Gordon Conference on Theoretical Foundations of Design and Manufacturing*, Plymouth NH, June, 2000.
8. Sowa, J. (2000) Knowledge Representation: Logical, Philosophical and Computational Foundations, F. Brooks/Cole Thompson Learning, New York.
9. Mills, J.J., J. Goossenaerts and H.J. Pels (2001) The Role of Context in the Product Realisation Process, *Proc. of the International CIRP Design Seminar*, held in Stockholm, Sweden, June 6-8, 2001.