Engineering with Computers (1995) 11: 167–172
© 1995 Springer-Verlag London Limited

Engineering with Computers

Modeling Evolving Product Data for Concurrent Engineering

K. J. Cleetus

Concurrent Engineering Research Center (CERC), West Virginia University, Morgantown, WV, USA

Abstract. *In order to make the traditional product structure tree representation amenable to concurrent engineering relationships like* perspective-of *and* dependent-on *have to be added to the essential* part-of *relationship. Complex data can be held in proprietary formats, while simple data will be in a common representation for direct access by diverse disciplines. Coordination among team members in a project can be carried out using such a model. Besides, a virtually unified view of all the data is possible, though they may lie in distributed and heterogeneous data bases. A very necessary characteristic of such a model is that its time evolution should be easy to represent in order to reflect the dynamic nature of product development, where the model itself, and not merely the data values change. Managing versions is also facilitated by the comprehensive structure of the Unified Product Data Model (UPDM).*

Keywords. Concurrent engineering; Configuration management; Data model; Design Process; Engineering data management; Product data management.

1. Introduction

Product data modeling has been extensively studied, focusing on the standards for expressing the data required to define the finished product for design and manufacturing (1). In the context of the DICE (DARPA Initiative in Concurrent Engineering) project these standards are seen as highly desirable to be adopted and followed by the CAD vendors and tool developers, so that part data of all types may be exchanged among the gamut of engineering tools. But DICE is not focused on the standards issues, but on the question of how best to share the product information among a team. Given that the engineering data are not represented in any substantial way today in universally recognized computer representations and file formats, the challenge is to see how far one can go without those standards in making information, resident in incompatible formats and possibly distributed, available, nevertheless, at the workplace of a product development team member without delay. Today's multi-functional product data are captured in proprietary formats and follow different representations. If one may hazard a guess, this is likely to remain the situation far into the future, because the tools themselves are being developed much faster than the standards. A second important requirement is that the data sharing in product development must concern not just the finished design, but also the evolving incomplete design which is constantly changing throughout the project.

The data that need to be shared fall into two classes:

- Extremely complex data structures containing hundreds and thousands of data elements with hundreds and thousands of relationships among them. Example: an electronic circuit layout schematic.
- Simple data consisting of numbers and text that characterize a part.

Both types need to be shared. The former type will be produced by an engineering tool in a proprietary format and need to be processed by another tool in its own proprietary format to conduct an analysis in another perspective of the total product design. The only known solution to this situation is to write custom translators to perform as faithful a computer translation as possible, and touch it up by hand finally.

The second type of information needs to be held in one common representation in order that humans and computer group work tools (such as constraint management systems and notification systems to coordinate team work) may directly access those data.

Correspondence and offprint requests to: K. J. Cleetus, Concurrent Engineering Research Centre (CERC), West Viriginia University, PO Box 6506, Morgantown, WV, USA. E-mail: jocle@cerc.wvu.edu

2. Unified Product Data Model

The Unified Product Data Model (UPDM) that we propose should hold the first type of information as a pointer to a file of a certain type; and the second type of information directly in computer memory, visualizable by humans, and programmatically accessible by computer tools. Assume that this information is placed there by one perspective; then any other perspective can bring the required *Complex* data into its own tool by invoking a translator on the file pointed to (the pointer is not the pathname of the file but a unique ID of the file which is translated to a pathname by an information server). A person can directly view the *Simple* data or they may be conveyed programmatically to any CE tool that processes the data (for example, a CE Notification System or a CE Constraint Management System). The name-space of the tool desiring the data and the name-space of the CE service that holds the UPDM will be different. Therefore all such accesses to the UPDM must be mediated by a look-up table to translate between the name-spaces. These tables should be embedded in the client tool.

The UPDM was suggested on p. 78 et seq. in the DICE Red Book [2]. It is built as a product structure tree with the basic *part-of* relationship to express the successively lower levels of decomposition of the product. The CE aspect of the data model consists of the fact that any individual node of the tree representing an assembly, sub-assembly, or component has multiple slots to represent the various *perspectives* of that node, and when one of the perspective slots is opened up it may consist of some Complex structures (pointers to proprietary file formats) and some Simple structures (a vector of numbers and strings). An example, relying as always on the now familiar turbine blade in DICE, is shown in Fig. 1.

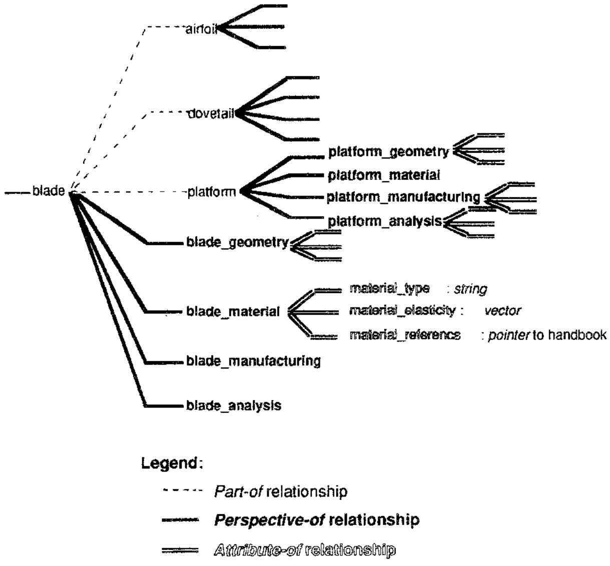

Fig. 1. Example of a Unified Product Data Model of a turbine blade in an aircraft engine.

3. Hierarchical Relationships

The *part-of* relationship defines the outline of the whole tree. The *perspective-of* relationship is a bag to hold all the data that are owned by and defined by one perspective (though others may have to assent consensually, and could be dependent on the data). The mutual influence of decisions on other decisions is also an essential component of a model of the product data to support CE. Hence all data attributes, whether Complex or Simple, have *dependent-on* relationships specifying perspectives that need to be notified if the value of that data attribute changes. In case certain automatic computing processes should be invoked, the *dependent-on* relationship performs that action too.

The hierarchical structure of the UPDM is excellent for encouraging the re-use of earlier product data. For example, an entire sub-assembly available from an earlier project may be attached to a node of the new product structure tree to record a design decision to use a previously designed sub-assembly. One may then modify a portion of it for the present project. Since industrial products, such as automobiles, may have only 10–15% new components from one model to the next, this re-use of earlier components, sub-assemblies and assemblies is the rule rather than the exception. The modeling methodology proposed above supports this effectively and graphically – and it is the intent to explore how it can be implemented in at least one of our data modeling tools, the Team Coordination enabling technology, called the Project Coordination Board [3].

4. Product Data Access

The UPDM is a virtual unification of all the data; it is not a common representation of all the data, nor does it attempt to be one. Common representations are extremely unlikely to be realized if one truly wishes to cover all the data that specify a product, from the diagnostic program to test a component in the field (logistics perspective) to the numerical control program that will be used to cut a mechanical part on a machine tool (manufacturing perspective), the multimedia user manual (documentation perspective), the shape (aerodynamic perspective), temperature profile (thermal perspective), and so on. One may even argue: what need is there to have a common representation when what one really needs are translators? The STandard for the Exchange of Product model data (STEP) is being developed as an international standard for product data sharing [1]. If there actually is one gigantic STEP model of all of the data to define a product, then the word *Translator* will be replaced by the word *Application Protocol.* Conceptually the world without STEP and the world with the most complete STEP are no different: when you need data to be exchanged between perspectives, you will have to execute a procedure in both cases.

The UPDM is also a directory to all the data that arise in the course of the product development and are needed *in toto* to define the product for all stages of the development, and for all perspectives that work with the product. In a UPDM for CE some of the data are really there, explicitly modeled and stored in computer memory and on auxiliary storage. Some of the data are present with one level of indirection, but still on-line, as an index to a data base or document repository – but the user has to switch context to get at that data. And some of the data are not present on the computer at all, because they are lying in a file cabinet and the pointer from the UPDM is a reference to a file folder within a file cabinet – a human must go and look it up to access it.

5. Process

The process by which the product is developed is not stored in the UPDM. It is captured as a set of reusable development processes, stored in a process library, and categorized by perspective. It is also stored in the history of the project as fragments of rationale and annotation of decisions taken. It is optional whether to consider these as Product Data which must be linked to the UPDM, or as Project Data, unlinked to it. The generality of the UPDM described above would indicate it is better to link the annotations in a design notebook as an annotation pointer (index) from the relevant node in the product structure tree into the Design Notebook data base [4].

The process also has the limited, but often used, connotation of the manufacturing process. This is indeed stored in the UPDM, because the UPDM contains the data required to define the product in the manufacturing perspective. Therefore the manufacturing perspective will have contributed to fully defining, after debating if need be with other perspectives, the final decisions on the manufacturing process. For example, it will define a tool, a jig, a tool path, a feed rate, etc., for a metal cutting process. So in the sense of manufacturing process, the product model does contain the process; though not in the sense of the product development process, which is considered in the previous paragraph.

6. Configuration Management

The product structure tree, for all the convenience it affords, is also a fairly compact way of holding the configuration of a product. Consider a product like an automobile with several thousand parts. Assume each part has half a dozen perspectives, and the simple data aggregates for each perspective, such as a length, a mass, certain elastic constants, etc., can be held in, say, 20 numbers, and the complex structures occupy at most half a dozen CAD files to which the product structure tree will only hold pointers. Including overhead the whole product structure tree will occupy just 2.5 Mb of data. Of course, the CAD files themselves may occupy a thousand times more data, gigabytes in extent, but that is not unexpected. The compactness of the product structure tree here advocated comes through when it is realized that the next model will need just another 2.5 Mb to hold the configuration, and an additional 15% or so for the CAD files of those parts that have actually changed from the earlier model. Because of the compactness it is quite possible in a project to hold even a score of versions at any time that are under investigation by different groups. However this proliferation is not expected to multiply geometrically in the nature of the basic concurrent engineering of product development, which enforces synchronization and consistency among various perspectives periodically [5]. This consensus is the basis of the sign-off through which the different perspectives agree and make consistent their respective contributions to product definition, so that for instance, the maintenance diagnostic provided by the logistics perspective for a part conforms to the functional schematic of the part put out by the design perspective. Hence the entire system of engineering change notices can be integrated with the product structure tree as a design annotation perspective with slots to embed the customary annotation for engineering changes; the actual engineering-intensive drawings describing the changed part are automatically pointed to from the product structure tree.

Once the changes have been made and agreed upon, those portions of the tree can be made read-only by the project leader to prevent further inadvertent change. However, during product development, before partial consensus is reached, it is likely that the values of several parameters will be changed by the responsible developers many times. The tree should have the flexibility of holding lists of alternate values with identification of the proposers. This many-valuedness will enhance the ability to represent the flux of activity in product development by providing the temporary storage needed to hold the input for studies of trade-off among different perspectives. There is a great deal of communication and interchange that goes on among the developers, quite apart from their coordination via the shared data of the product structure tree. It is the purpose here to advocate the extreme usefulness of sharing the evolving product data model as a foundation for coordination among product developers; but there is even more coordination that needs to take place via meetings, messages, and multi-media communication.

It is clear that the UPDM for a finished product can provide a very essential function of structured Configuration Management. The tree has all the information (or pointers to the information), in every form, that defines the product. An NC program to machine a component on a machine tool is there, pointed to from the Manufacturing perspective for that part. A stress analysis color plot for the part is also there, pointed to from the analysis perspective. A service manual for the product is also present, referenced from the logistics perspective. And so on, for all other artifacts, documents, computer programs, jigs, drawings, parts lists, etc., which constitute the product definition. The files themselves may be on-line to the user, or off-line managed by other computer systems, or by humans tending hard copy archives. In the ideal case all the files would be in an electronic store, at minimum scanned in from hard copy, and all of them would be accessible on-line to the authorized personnel; but it is the UPDM's comprehensive structure that makes it possible to define a Product Version uniquely and completely, and the UPDM at least will be available on-line. Product developers will be able to know from their workplace the *entire* information constituting the product or any component of it, no matter where it is resident in the organization and in what form. This in itself is a vast improvement over the situation prevailing in industry today.

Consider the need to sub-contract the manufacture of a particular part. The UPDM will lead one straight to the part and reveal the total information available defining the part, and any required drawings, annotation, documents, materials, jigs, etc., can be marked for printing with a utility. The precise documents and drawings needed, organized in a meaningful sequence for external manufacture, are thereby available in short order. These so-called technical data packages (TDPs) are extremely hard to get at and retrieve in organizations that carry a great variety of products [6].

7. CITIS and CALS

This aspect of the UPDM parallels the goals of the Contractor Integrated Technical Information Services (CITIS). CITIS is meant to be the single entry point for access to technical data stored at the site of a contractor, who is obligated by a government contract to supply electronically the technical data specified in a contract for acquisition of a product from that vendor [7]. Some contracts call for no more than a list of items to be available on-line to the government, but they may go so far as to mandate access to the actual CAD data and its translation into an agreed neutral format. The UPDM is suited to this purpose admirably. By browsing the product structure tree, at its appropriate level of abstraction, the user can identify the data that are available, what it pertains to, and in what native form it exists. If the user is interested in accessing the data, provision has to be made for file transfer and translation after performing user access controls. More often than not, the user wishes to view the data on the screen; one can safely assume the frequency of the view type of accesses will far outnumber the file transfer type of accesses. What can be done to satisfy this critical need which was identified on p. 90 of the Red Book [2]?

Normally the model and the accompanying data dictionary are considered as meta-data, i.e. data that describe the definition, structure, form, purpose, units, etc., of the data stored about the artifact. We propose that such meta-data be extended to include access methods to obtain the artifact data. The UPDM should hold pointers to the necessary software routines for accessing the data lying in remote files. Further, this should also enable a user to execute the proprietary applications on remote computers using those files and bring the resulting graphic image to the workstation from which the user is currently interrogating the UPDM. The methods may be recorded explicitly at the time the data are published on the UPDM by the product developer, or they could even be generated automatically from the meta-data. In this way the goal of heterogeneous distributed access to all of the data comprising the product can be achieved in a uniform way. Moreover, such data access will be just as usable and advantageous during product development (the focus of CE), as it is after product development is completed (the focus of CALS — Computer-aided Acquisition and Logistics Support — and CITIS). Indeed, in the sense that CE visualizes the development cycle as continuing until the product's retirement from use, the CALS requirement of data access may be subsumed in the CE requirement.

8. Model Itself Changes

The UPDM is meant to support the product development process, which goes through several distinct stages in every industry. At each stage, the type of decisions that are taken are different, and the UPDM needs to support the collaboration of persons on those decisions. The UPDM is a skeleton to hold partial decisions around which the team's work will be communicated and coordinated. It is on these decisions that there must take place the propagation of influence of one decision on another, in order to ultimately reach consensus. We posit that not one, but several UPDMs are needed over the product development life-cycle. The one at the concept development stage will address the requirements issues heavily, as well as the target performance parameters, the shape, the base technologies that will be used, and the market projections and risk. Further down the road, when one of several alternatives for the concept has been chosen, the UPDM should encourage the visualization of the entire development process that must be undergone, with possible iterative cycles, in many perspectives. This will therefore be a different model that will show up the architectural composition of the product and key starting points for the design in various areas. The final product model at the end will be the fully fleshed out model that was adumbrated in Fig. 1 above, one in which all the necessary product design, manufacturing, and logistics characteristics and support procedures have been identified and developed. In sum, the UPDM for CE should really be a family of UPDMs, one for each distinctive stage of a product's development cycle. Furthermore, the UPDM is a dynamic, changing, and incrementally evolving one as the product is developed.

This dynamism is part of the normal IPD (Integrated Product Development) process. Hence, it is inappropriate to define models that are *compiled*, as though all programs that access the model thereafter are going to be working from a constant and unchanging model. At any point, any perspective has the latitude to freely add to the product structure tree — new part-of nodes, new perspectives, new attributes — and without this capability the whole IPD process would be hobbled, and would not reflect the great flux and creativity that accompanies team involvement in product development. It is this fact that makes any modeling process that compiles some model written in a computer language into an invariant schema, wholly unsuitable for the *product development process*. The schema should be built up interactively and graphically with a tool that supports the building of schema trees depicted above; and the engineering user should have the ability to import interactively into his program (by a programmatic interface), or write to a file in a standard format (by a programmatic or user interface), any desired portion of the whole tree. A computer data modeling tool satisfying the above critical requirements exists in one enabling technology at the Concurrent Engineering Research Center (Team Coordination) and will be used consistently with another enabling technology (the Information Sharing Server). The interchange between them, as also the exchange between data held in these CE services and any CAD tools, will take place through a single intermediate file format, conforming to an acceptable standard that other vendors can also be expected to inter-operate with. This format has not yet been decided, but is likely to be a STEP physical file format with its accompanying EXPRESS schema generated on the fly to reflect the current state of the product definition. It will be natural to generate EXPRESS definitions from the internal representation of the dynamic model, as a way of conveying the model to any other tool in a standard way, and this should be possible within the limitations of defining arbitrary data types and entities in EXPRESS.

9. Dynamic vs Static Data Model

The creation and evolution of such a dynamic data model are central to the coordination of the team because the data model has precisely the data elements around which cooperative decisions are to be taken at that stage of the product development. It is held in the memory of the Team Coordination tool called the PCB (Project Coordination Board) [3], and constraints, notifications, and ownership links are attached to the explicitly held data therein (the simple structures). Whenever a data attribute value changes it is these links that enable the evaluation of constraints and cause the notification of the team members concerned. Therefore, it is a unique feature of the UPDM for CE that it holds that type of data too – who is affected by a data attribute value, and what requirement of the customer or limitation imposed by the organization must be checked when the value changes.

The product data model that will be built by the ISS (Information Sharing Server) [8] at CERC will serve to model the well-defined and completely laid out information of past products that is contained in some commercial data bases. Naturally, this data model is static; and one may separate the process of data definition of the *class* and data population of the *instances*. The ISS serves as the uniform gateway to access past information about numerous products contained in legacy data bases. The UPDM, by contrast, is meant to support the inchoate, still incomplete, and constantly evolving definition of a new product that is under development. The UPDM does not separate data definition from instance data assignment and storage; it is a structure through which one specifies at once the instance data of a single new product, as well as its structure.

10. Summary

To summarize: the unified product data model is a dynamic product structure tree linked by the *part-of* and *attribute-of* relationships, which are amplified for CE by the essential *perspective-of*, *owned-by* and *dependent-on* relationships. They allow complex data in proprietary file formats to be pointed to and exchanged by translation steps, or simpler data (aggregates of numeric data) to be held explicitly in a common representation in computer memory and exchanged by direct programmatic access or visual access with a graphical data visualization tool. The Team Coordination and Information Sharing Systems will agree on a common format for the physical data file and corresponding data definition to exchange data. The UPDM is a virtual unification of all the data; it is not a common representation of all the data. The underlying tree of the UPDM is a key to configuration management also. A family of qualitatively different models are necessary at different stages of product development. Important CE services can work from this unified data model: the team coordination service, the information sharing service, the constraint management service, the design annotation service, and the methods for integration of CAD tools with CE services. The UPDM is therefore the most important element in constructing a CE environment, but its propensity to change necessitates using an interactive graphical tool to accomplish the modifications. However, at any time, one may extract the current data definition (schema) in a standard descriptive modeling language, such as EXPRESS, and the current data in a standard format, such as the physical STEP file fornat.

Acknowledgements

I am grateful for a long conversation with Dr V. Jagannathan, Dr R. Raman and Dr G. Trapp, which clarified some of the issues. I also wish to record my appreciation to Mr R. Bryant of the Raytheon Company for a demonstration of one of their internal engineering data management systems called RITIS.

This work has been sponsored by the Defense Advanced Research Projects Agency under Grant MDA-972-91-J-1022 for the DARPA Initiative in Concurrent Engineering (DICE).

References

1. Carver, G.P.; Bloom, H.M. (May 1991) Concurrent Engineering Through Product Data Standards, NISTIR 4573, National Institute of Standards and Technology.
2. 2. Cleetus, K.J.; Uejio, W.H. (Eds) (Feb., 1989) Red Book of Functional Specifications for the DICE Architecture, Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.
3. Londono, F.; Cleetus, K.J.; Nichols, D.M.; Iyer, S., Karandikar, H.M.; Reddy, S.M.; Potnis, S.M.; Massey, B.; Reddy, A.L.N.; Ganti, V. (April 1992) Managing Chaos: Coordinating a Virtual Team. Proceedings of the First Workshop on Enabling Technologies for Concurrent Engineering. Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.
4. Uejio, W.H.; Carmody, S.; Ross, B. (June 1991) An Electronic Project Notebook from the Electronic Design Notebook (EDN), Proceedings of the Third National Symposium on Concurrent Engineering, Society for Computer Aided Engineering, Rockford, IL, and Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.
5. Cleetus, K.J.; Reddy, R. (June 1992) Concurrent Engineering Transactions, Proceedings of the Fourth Annual National Symposium on Concurrent Engineering, Society for Computer Aided Engineering, Rockford, IL, and Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.
6. Nomura Enterprise Inc. (Feb., 1992) Next Generation Engineering Data Management Phase II Report, prepared for US Army Industrial Engineering Activity, Rock Island, IL.
7. Stickman, J.F.; Rosenthal, H.G. (1992) CITIS: Gateway to Defense Industrial Base Productivity, CALS Journal, 1, 3, Fall.
8. Karinthi, R.; Jagannathan, V.; Montan, V.; Petro, J.; Sobolewski, M.; Raman, R.; Trapp, G.; Deng, S.; Almasi, G.; Li, X. (Aug., 1993) Modeling Enterprise Information and Enabling Access using the Information Sharing Server, Proceedings of the Seventh Annual Engineering Database Symposium, San Diego, CA.