7th Workshop on Product Structuring – Product Platform Development
Chalmers University of Technology, Göteborg, March 24-25, 2004

RELATIONSHIPS IN PRODUCT STRUCTURES

Alison McKay, Damian N.H. Hagger, Charles W. Dement, Alan de Pennington, Peter Simons

Abstract

A number of factors are driving today's product development processes. Increasing demands for sustainable development and concern for whole-life costs are changing the emphasis from product realisation to product life-cycle engineering. Competition is forcing companies to redefine their goals and the business models that deliver them. Consumer demand for bespoke products is leading to mass customisation and the adoption of product postponement strategies. As a result, new types of supply chain structure and life-cycle process are emerging. These are likely to continue evolving as new computational infrastructures, such as the Grid[1], become more widely used.

These changes are placing increasing pressures on the information systems that are used throughout the product life-cycle. It is no longer sufficient for the product model to support just the design and manufacturing processes – it must support the entire product life-cycle, including processes that have yet to be defined. It is crucial that today's process structures are not inadvertently built into the information systems that will be used 20-30 years from now. The question of how to define a product, and the platform upon which it is built, in such a way that it will facilitate life-cycle processes – some of which may be many years away and whose definition is not yet determined – is a key driver behind the research reported in this paper.

This paper reports research that has resulted in the identification and definition of a number of different types of relationship that occur in product structures. Product data representation schemes embodying these different kinds of relationship have been built and evaluated with a range of product structures from a number of industrial sectors. In parallel, a Grid-enabled product structure viewer (built upon this framework) has been built. Experiences gained from early attempts to view product structures on the Grid[1] will be discussed.

1 Introduction

Currently there is no adequate theory upon which product data representation schemes can be based. Such representation schemes underpin product, and so product platform, life-cycle models. Shortcomings in product data representation can have serious consequences for the usefulness and effectiveness of the information systems that they underpin. Product platform life-cycle models are underpinned by product data representation schemes that include a

[1] A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [1][Foster et al, 1998]; the Grid (and Grid computing) is concerned with the delivery of coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [2][Foster et al, 2001].

range of product structures that cover the life-cycle of the platform and the products that are built upon it. If the models are to provide effective support through the life-cycle of the platform and products then the underlying representation schemes must support the entire life-cycle and the range of computational infrastructures that are used to support life-cycle engineering processes. In addition, product structure visualisation and simulation methods and tools will be required.

2 Background

The product definitions that support life-cycle processes result from product design processes. Product design is inevitably a multi-disciplinary endeavour and different disciplines have different viewpoints on what constitutes a design process. For example, engineering and science-based design research tends to be conducted against a backdrop of a well defined design process (of which there are many, for example, [3, 4]) whereas researchers from the visual arts and humanities tend to regard such processes as inhibitors to creativity and innovation and are far less prescriptive [5]. Design practitioners, on the other hand, work within stage gated business processes [6]. They need design methods and tools, for example to allow systematic evaluation of design alternatives at decision gates, information support systems to allow the digital definition of designs and data needed for downstream life-cycle processes and they need to work in environments that foster both creativity and innovation. The research reported in this paper focuses on a key aspect of any product design process, namely, the information that it uses and produces and the different ways in which product definitions are represented.

It is not feasible to assume that a single product data representation scheme for all products and processes can be defined and used in all situations. A more realistic scenario is that specific representation schemes will be designed for specific purposes. In this context, frameworks for product data representation are needed to guide the implementation and deployment of product information systems. A number of such frameworks are available. McKay et al propose a framework for electro-mechanical product definition data that decomposes the product definition with respect to level of detail and life-cycle phase [7]. Van den Hamer and Lepoeter [8], on the other hand, identify five dimensions of product definition data that occur in product data management systems which support configuration management processes. More broadly, the STEP integrated resources offer what can be seen as a framework for the definition of products in general [9]. All of these frameworks are open to the criticism that they lack a well-founded theoretical basis. Recent work related to design ontologies, for example as a part of engineering design science research [10], aims to address this shortcoming. Research on the representation of product structures has led to a framework for product structuring that enables the representation of product, process and organisational structures as collections of elements and relationships [11,12]. The kinds of element and relationship to be captured vary according to the context within which the product structures are to be used. For example, a production planning activity may need a product breakdown structure (parts related by composition relationships) whereas an assembly planning activity may need a physical assembly structure (parts related by mating condition relationships). The research reported in this paper has resulted in the identification and definition of a number of different types of relationship that occur in product structures.

A number of software prototypes have been built upon product data representation schemes that embody the different types of relationship. Each prototype has been evaluated with sample data. This paper will report on an application of the most recent software prototype –

a Grid-enabled product data (Bill of Materials) viewer that is intended for use to support product maintenance processes. The Grid can be regarded as an evolution of the World Wide Web. A key difference is that the resources available on the Web are information whereas the Grid allows access to a range of different types of resource including information, application programmes (including engineering databases) and computer processing cycles. The earliest Grid implementations are allowing scientists and engineers to run high-performance computing applications, with large volumes of data, on high-performance computers that are accessible remotely from themselves through the Grid. Grid resources are made available to users through Grid services. A Grid service is, in essence, an extension of a web service. The Bill of Materials viewer was implemented as a Grid service and linked to an engineering database, containing Bill of Materials type structures (parts related through composition relationships), that was hosted as a web service. The Grid service allows users who are remote from the database to view product structures stored in the database.

3 Relationships in product structures

Three categories of product definition data are presented:

- the description of a product at a point in its life-cycle and time;
- information needed to support configuration management type activities;
- relationships needed for product realisation.

The presence of each of these kinds of product definition data governs the extent to which a given information system can support product life-cycle processes.

3.1 Relationships needed to describe a product at a point in life-cycle and time

If the goal is to capture a given product structure at a given point in time then only core relationships are needed. These are summarised in Table 1.

Table 1. Core Product Structure Relationships

	Type	**Relation of**	**between**
Core	composition	containment	Entities <-> elements [of entities]
	constitution	materiality	Entities <-> materials
	inherence	characterisation	Entities <-> attributes
	qualification	conditionality	Relations <-> antecedents
	quantification	multeity	Relata <-> quantities

3.1.1 Composition relationships

Composition relationships are the part-whole relationships in Bill of Materials (BOM) type product structures. From the branch of philosophy called mereology, composition relationships can be either integral or distributive [13].

- For integral composition relationships, at some degree of abstraction, parts are the same kinds of thing as the whole and all of the parts are spatially contiguous. For example, an

assembly can be said to contain parts; parts may be either components or assemblies; thus the parts of are an assembly are the same kind of thing as an assembly itself. This aligns with the composition relationships defined in the framework for product structuring [11].

- For distributive composition relationships, parts are not necessarily the same kinds of thing as the whole and the parts are not spatially contiguous. For example, the parts (components and assemblies) of a brake system (a system that delivers braking functionality) on a car are not physically co-located but together serve the functions that are designated by the name "brake system".

3.1.2 Constitution relationships

Constitution relationships identify the medium through which an entity is realised: for example, the type of material from which a part is made. A given entity can have only one constitution. The entity to which a constitution relationship applies can be either dependent or independent.

- For dependent entities, the constitution of the entity is governed by the constitution of its parts (which can be established from its composition relationships). For example, the constitution of an assembly is the materials of its component parts plus any materials used to form the assembly.
- For independent entities, the constitution relationship identifies the constitution of the entity directly, for example, the material of a component.

3.1.3 Inherence relationships

Inherence relationships relate entities to their properties. A property is an intrinsic characteristic of an entity that is ontologically distinct. For example, the hardness property of a material is ontologically distinct from the colour of a material.

Given that both elements and relationships are parts of a product [14], both elements and relationships in a given product structure can be characterised. For example, mating conditions might characterise an assembly relationship whereas shape might characterise a component part. According to the STEP standard [15], properties are defined, represented and presented – these are beyond the scope of this paper. Inherence relationships themselves can be either facetted or dispositional:

- Facetted inherence relationships are those between an entity and its spatial features, for example, topological and geometric forms such as holes, protrusions, faces and surface deformations.
- Dispositional inherence relationships, on the other hand, are those between an entity and any property or quantity with a determinable (but not necessarily quantifiable) magnitude.

3.1.4 Qualification relationships

Qualification relationships relate [product structure] relationships with conditions that govern their existence. For example, the definition of a car body has two revisions; Revision A has a hole for a sunroof in it whereas Revision B does not. The fact that the car body includes a hole for a sunroof would be established through the inclusion of a facetted inherence relationship between the body and the hole in the definition of the car body. A qualification relationship would then be applied to the inherence relationship to specify that the relationship exists in Revision A of the car body and that it does not exist in Revision B of the car body.

In practice, general purpose industry strength implementations of qualification relationships are not achievable. A key issue is that such an implementation would require computer processing capabilities linked to the product data representation scheme itself, rather than the programming language that is used to express it. A particular challenge is the realisation of algorithms that support the full range of arithmetic operators applied to representations of physical quantities that are richer than a simple number or numeric expression.

3.1.5 Quantification relationships

Quantification relationships relate [product structure] relationships with physical quantities. For example, a quantification relationship could be used to say how many engines were on a given aeroplane. The definition of a quantification relationship and its associated physical quantities has a number of parameters.

Quantity: designates a particular number of units.

Unit: designates a standard quantity in terms of which the quantity is expressed.

System: defines the system of measurement from which the unit is taken.

Note: The STEP measure_schema [9] supports the concepts of quantity and unit with reference to the ISO system of measurement [16].

Magnitude: designates the kind of physical quantity with reference to its dimensionality [17].

Datum: designates the entities that fall within the scope of the quantification.

For example, a length of 5mm would have a quantity of "5", a unit of "mm", a system of "SI" and a magnitude of "length". The value of a datum cannot be illustrated through this example because the idea of a datum covers multiple physical quantities.

3.1.6 Designation

Designation relationships relate entities with names or codes that are used for identification purposes. For example, a designation relationship could be used to assign a part number to a part.

3.2 Relationships needed to support configuration management

Section 3.1 covered the core relationships needed to capture product structures at a given point in time and in the product life for a given product. In this section an extra dimension is added – those needed to support configuration management type activities. For this, intrinsic adjunct relationships are needed: equivalence, alternation, variation, order and transformation. Each relates entities to other entities.

Before continuing further, some brief notes on the terminology used in this paper are provided. These notes are taken from a previous paper [18].

A product family identifies the commonalities and differences between the individual products that form a product range. The range of standard hexagonal metric nuts is a good example. The product family defines the commonality of all such nuts (for example, the fact that they are hexagonal with a threaded hole) with the differences between them (for example, the dimensions) given as parameters. A variant of a product family is an individual product that conforms to the product family. For example, an M6 nut is a variant of the standard hexagonal metric nut family. It has all of the features that are common to the family and parameter values that are specific to itself, for example, the 6mm diameter. A range of

products is a set of variants of a single product family. For example, M6, M8, M10 and M12 nuts are a range of standard hexagonal metric nuts.

The variants of a product family are not versions; rather they are different potential products that belong to the family, each serves a different purpose. Different versions of a given product result from change; each version of a given product serves the same purpose but in different ways. For example, a later version of a given product may be an improvement on a previous version.

3.2.1 Equivalence relationship

Equivalence relationships relate [product structure] relationships with things that can act as substitutes for the dependent thing in the relationship. For example, an equivalence relationship might be used to define a substitute part in a composition relationship in a BOM or it might be used to define a substitute constitution of a given entity.

3.2.2 Alternation relationship

Alternation relationships capture options. Each alternate in a given group has a different functionality but is satisfactory in the context of the thing within which the alternates are acceptable.

3.2.3 Variation relationship

Variation relationships represent diversity. They define relationships between baselines (for example, the generic BOM of a product family) and variants of the family.

3.2.4 Order relationship

Order relationships are used to define the relative positions of entities with respect to each other. For example, a process plan has an ordered sequence of steps to define, for example, an assembly process.

3.2.5 Transformation relationship

Transformation relationships show development over time. They capture states of entities, for example, the versions of a product.

3.3 Relationships needed to support entity realisation

The kinds of relationship defined so far are required for product definition processes. Three additional relationships are needed for product definitions needed to support manufacturing processes: articulation, factorisation and consolidation. Collectively these are referred to as extrinsic relationships. These relationships relate product structures, rather than elements and relationships, to each other.

3.3.1 Articulation

Articulation relationships allow physical entities to be disaggregated. For example, a disassembly process needs a product structure that defines the initial assembly and at least two product structures that define the disassembled pieces.

3.3.2 Factorisation

Factorisation relationships allow the physical combination of product structures. For example, sheet metal parts might be nested when cut from a sheet of material; factorization allows the conglomeration of a number of parts to be defined.

3.3.3 Consolidation

Consolidation relationships allow physical entities to be aggregated, for example, in assembly processes.

4 Grid enabled product structure viewer

The Grid enabled product structure viewer supports the visualisation of composition relationships in product structures.

Distributed computing is a way of using many computers to perform a single computing task. Umar [19] has proposed that a distributed computing system is "a collection of autonomous computers interconnected through a communication network in order to achieve business functions''. An advantage of using distributed computing is that advanced computing and storage capabilities can be integrated into a single application "system". This system can be constructed by breaking software into components that can be mapped to resources on high-performance networks, with each application component taking advantage of unique capabilities such as fast storage, vector processors, and massively parallel systems [20].

Recent research efforts into decentralised distributed computing have focused on the Grid, a paradigm that has been defined as "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" [2].

The demand for collaborative software environments with heterogeneous, loosely coupled computer resources, sophisticated workflow management capabilities and built using service-oriented architectures is growing. Such environments involve the sharing of both information and processors that operate upon this information. Research into Grid technology is aiming to build such collaborative environments and Grid technologies are at an early stage of development.

Service oriented architectures, such as the Grid, are unlike distributed information system technologies, such as Common Object Resource Broker Architecture or Distributed Component Object Model, because they do not rely on tight coupling between resources or vendor-specific implementations. This relieves the scalability and process interoperability problems that frequently arise when distributed information system technologies are deployed. Current Grid implementations are using open standards and are not specific to a given vendor.

The building of a Grid-based BoM viewer has been used as a case study to implement one of the relationship types within a Web service. A product structure database was encapsulated as a Web service hosted on an Apache Axis server (Figure 1). A second Web service was created to construct the composition relationship visualisation, by accessing the database. An interface written using Microsoft .Net technology was used to access the visualisation Web service and to display the visualisation.

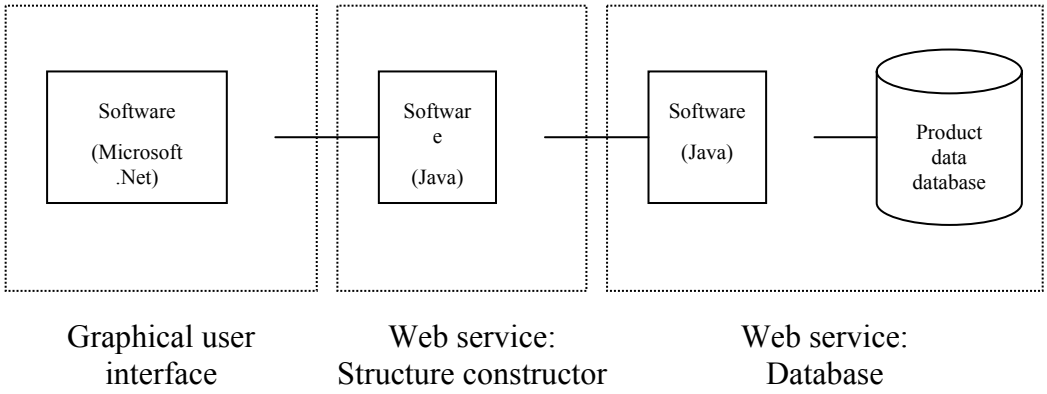

Figure 1. Web services: connections and content

The aim was to investigate the ability of Grid services to represent and visualise product structures that are more complex than currently achievable from disparate heterogeneous computer systems. Such computer systems require ad-hoc custom built solutions.

Hardwick [21] and others have researched the use of distributed [software] component technology to support the product lifecycle. This research differs in the underlying technology and the opportunity that it offers for outsourcing the means to implement an information system.

5 Results and conclusions

Three categories of relationship have been identified:

- those needed to describe a product at a given point in its life-cycle and time;
- those needed to support configuration management type activities which require relationships on how a given product (and so its structure) has changed over time; and
- those needed to describe how a product relates to a given product life-cycle process, such as manufacturing or maintenance, and the agents that execute it.

Each of these contains a limited number of types of relationship and governs the extent to which a given information system can support product life-cycle processes to which the product will be subjected. Key observations from the establishment of the Bill of Materials viewer are that the same problems with product data exchange and interoperability exist in Grid computing as in any information system; the Grid does not inherently solve any of them, in fact, it exasperates them.

Early indications suggest that the service oriented architecture of the (relatively) open system could allow the elevation of the semantics of the product out of the service and into the business workflow management layer of abstraction, a key characteristic of a Service Oriented Architecture. This may help in the alignment of product structures with the business processes and models of an organisation. It may be argued that CORBA can achieve an SOA, however it does not provide the same level of flexibility required for setting up a collaboration that can be achieved with Grid technology.

Further to this, there are economic value propositions that emerge. With no vendors, there is no fee for using the technology within an application. There is no need to change the integration method when one project ends and another begins if all projects use Grid standards. The research into Grid technology is still in its infancy, and industrial-strength implementations of it may be years away.

Composition relationships can be stored and visualised using Grid services and there are advantages to doing so:

- Databases storing data about relationships and the product can be used to construct a single representation of a product. The use of distributed databases across Grid services needs to be tested. Early efforts[2] indicate that the time taken to do an SQL SELECT statement across distributed Grid databases is considerably less than existing distributed databases.
- Services can be used to provide knowledge (in the form of software) required to construct a visualisation from a database.
- Engineering analyses can be programmed into Grid services which can be accessed through a graphical user interface.

There are two disadvantages found by the research

- Grid services do not inherently support product structures: they still require conformance to standards such as STEP
- Grid service software development requires skills beyond those required for homogeneous, fixed network solutions.

The research also raised the following considerations:

- Information systems designed to be implemented using Grid technology require extra dimensions to be considered e.g. the outsourcing of the processing capability leading to issues such as loss of control of the information system itself; intellectual property of the process owners being in the hands of process executors.
- Configuration of Grid services is required to allow fit-for-purpose information systems to be developed. This leads to the need to be able to represent the services themselves as parts, with relationships between them.

There are questions that still need answering regarding the role of Grid technology in product structuring and lifecycle support.

- Can the entire lifecycle of a product be supported by an information system implemented using Grid technology?
- What are the characteristics of the engineering problem that make it suitable to be analysed on the Grid?
- What engineering needs can be fulfilled by using the Grid?
- How can the collaborative needs of an organisation be fulfilled using the Grid?

[2] AstroGrid, Access Grid Lecture to White Rose Grid and NE e-Science Centre by Guy Rixon on 04/02/2004

6 Acknowledgements

The authors would like to acknowledge UK EPSRC/e-science programme for their support through the DAME project (Grant No. GR/R67668/01) and the Keyworth Institute, University of Leeds which has provided the interdisciplinary research environment within which research such as this is conducted.

7 References

[1] Foster I, Kesselman C., "The Grid: A Blueprint for a New Computing Infrastructure". 1998, Morgan Kaufmann

[2] Foster I., Kesselman C. Tuecke S., "The Anatomy of the Grid", International Journal of Supercomputer Applications, Vol. 15(3), 2001.

[3] G. Pahl and W. Beitz, "Engineering Design: A Systematic Approach", Springer-Verlag, 1988.

[4] S. Pugh, Total Design – "Integrated Methods for Successful Product Engineering", Addison-Wesley, 1990.

[5] Dominique Vinck (editor), "Everyday engineering: An Ethnography of Design and Innovation", The MIT Press, 2003.

[6] M. Weske, W. M. P. van der Aalst, and H. M. W. Verbeek, "Advances in business process management", Data & Knowledge Engineering, Volume 50, Issue 1, July 2004, 1-8.

[7] A. McKay, M. S. Bloor, and A. d. Pennington, "A Framework for Product Data", IEEE Transactions on Knowledge and Data Engineering, vol. 8, pp. 825-838, 1996.

[8] Peter van den Hamer and Kees Lepoeter, "Managing Design data: The five dimensions of CAD frameworks, configuration management and product data management", Proceedings of the IEEE, Vol 84, No 1, January 1996 p40-56.

[9] ISO 10303-41, "Integrated generic resources: fundamentals of product description and support", ISO 10303-41, 1994.

[10] Eekels J, "On the fundamentals of engineering design science: The geography of engineering design science. Part 1" JOURNAL OF ENGINEERING DESIGN, 11 (4): 377-397 DEC 2000

[11] McKay, A, Pennington, A de, Trott, SJ, "Relating Product and Process Structures", Design for Configuration: A debate based on the 5th WDK Workshop on Product Structuring, 2001, pp117-128, Series Editor: Riitahuhta, A; Pulkkinen, A

[12] Andreasen, M. M., Hansen, C. T., Mortensen, N. H., "On the identification of product structure laws", Proceedings of the 3rd WDK Workshop on Product Structuring, Delft University of Technology, Delft, The Netherlands, June 26-27, 1997. pp. 1-26.

[13] Simons P., "Parts: Study in Ontology", Clarendon Press, 2000

[14] McKay, A., de Pennington, A., "Towards an integrated description of product, process and supply chain", International Journal on Technology Management, Vol. 21(3/4), 2001, pp. 203-220

[15] ISO 10303-1. "Overview and fundamental principles", ISO 10303-1, 1994.

[16] ISO 1000, “SI units and recommendations for the use of their multiples and of certain other units”, ISO 1000, 1992

[17] ISO 31-0, “Quantities and Units: General Principles”, ISO 31-0, 1992.

[18] McKay A., Erens F., Bloor M.S., “Relating Product Definition and Product Variety”, Research in Engineering Design, 1996

[19] Umar A., “Application (Re)Engineering: Building Web-Based Applications and Dealing with Legacies, Prentice Hall: Upper Saddle River, NJ.

[20] Charlie Catlett, “Standards for Grid Computing: Global Grid Forum”, Journal of Grid Computing, 2003, pp.3-7

[21] Hardwick M., Morris K.C., Spooner D.L., Rando T., Denno P., “Lessons Learned Developing Protocols for the Industrial Virtual Enterprise”, Computer-Aided Design, Vol 32, 2000, pp.159–166

Damian N. H. Hagger
School of Mechanical Engineering
University of Leeds,
Leeds,
LS2 9JT
UK