Java Based Solution for Generic Product Data Browsing

Kari Kaitanen
VTT Building Technology, Espoo, Finland

ABSTRACT

In its' earlier and on-going research projects VTT has developed generic meta-level product data models considering especially the needs of the construction industry. In this particular approach a new application, generic product data browser, has been developed on top of this fundamental modelling work by using the new Java technology.

Java is a fully object-oriented platform independent programming language, which enables you to make easily robust, multi-threaded implementations fully integrated to Internet. The new technology supports also component based programming (JavaBeans, CORBA) and derived objects. The browser application, called "Starlet", is fully implemented with Java and can be run anywhere through any Java-supported web-browser like Netscape or MS Internet Explorer. The applicavpvption can be run also locally as a stand-alone program.

The product data model used by "Starlet" is based on a generic meta-model. This high-level schema, defined by EXPRESS-G, supports generic product data management and gives the application a lot of flexibility and scalability in describing your product data class structures and for example object grouping. The schema supports easier mapping to and from e.g. already available and yet to come IFC-schemata without any loss of product data information.

1. JAVA TECHNOLOGY

Java is a fully object-oriented platform independent programming language, which enables you to make easily robust, multi-threaded implementations fully integrated to Internet [Heller et al. 1997]. The new technology supports also component based programming (JavaBeans, CORBA) and derived objects.

Java technology is a very rapidly growing technology nowadays. However many things that you would expect to exist already in the other environments, like MFC (C++), can still not be implemented with Java. The standardisation does not yet fully support things like mouse dragging or other very useful things in good software development.

Many things can be however implemented in other, very often more difficult - and unfortunately non-standard - ways. In most of the times it would be better to just wait the standard to expand enough, but the process has still been quite slow. The new forecoming standard

of Java however does promise a lot and e.g. I/O methods that have been made very secure in the earlier versions is going to be handled in the future in a more open way.

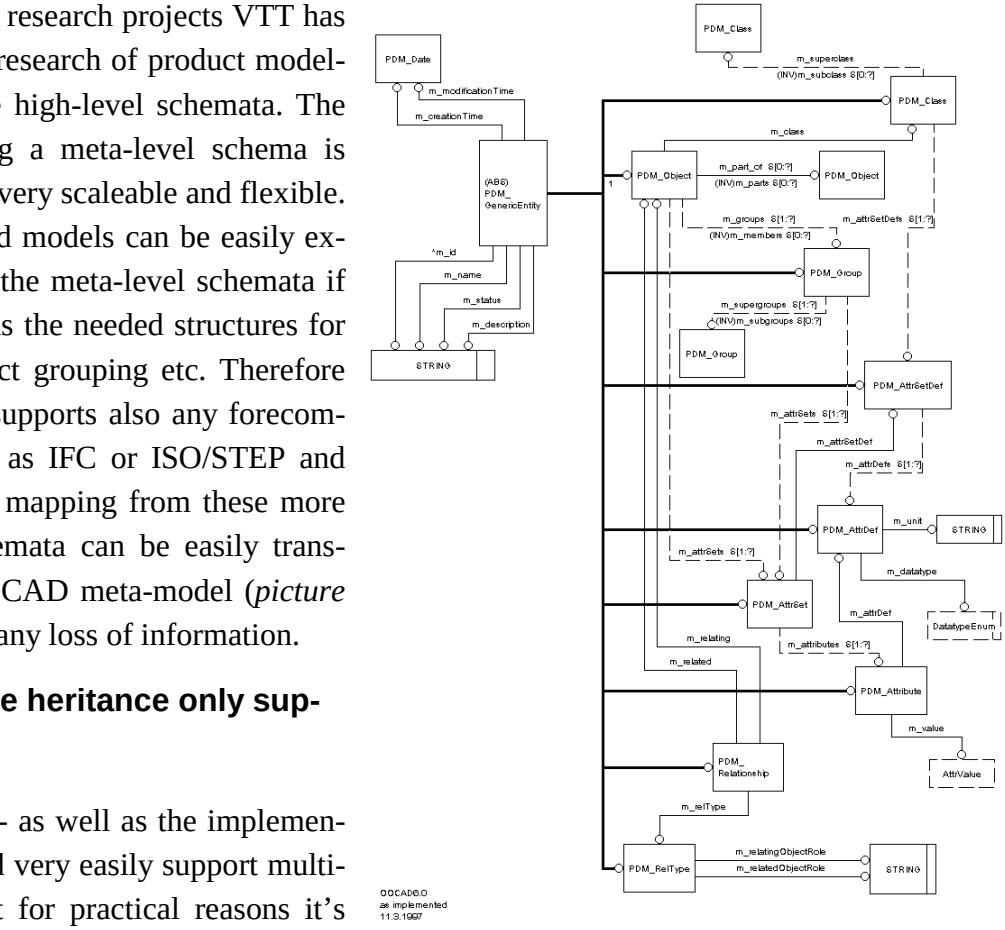
1.1 Limitations of current development tools

One major problem in implementing Java is that the Java based development tools are still a bit immature. In this approach VisualJ++ development tool was chosen. VisualJ++ behaved rather well and almost no crashes with the system or development tool was hampering the work. This is probably because the tool is mostly based on VisualC++ which has already got rid of the major child diseases it has had before. Of course, as Java is growing very fast, better versions of these tools are likely to appear while typing on this document.

One major point and a good example of deficiencies in the used development tool was the need of an outline box -control, which could be found in the control menu of the interface wizard (VisualC++) - but no Java code was yet generated for this control. As said these kind of weaknesses are of course disappearing very quickly as the tools are developed further.

2. USED META-MODEL SCHEMA - “OOCAD 6.0”

In its’ earlier research projects VTT has founded the research of product modelling on quite high-level schemata. The idea of using a meta-level schema is that they are very scaleable and flexible. More detailed models can be easily extended from the meta-level schemata if the model has the needed structures for flexible object grouping etc. Therefore the schema supports also any forecoming standard as IFC or ISO/STEP and product data mapping from these more detailed schemata can be easily transferred to OOCAD meta-model (*picture 2.1*) without any loss of information.



Picture 2.1. OOCAD 6.0 model - as implemented.

2.1.1 Single heritance only supported

The schema - as well as the implementation - could very easily support multiheritance but for practical reasons it’s not accepted. Multiheritance often leads to awkward problems in for example

method handling.

2.1.2 Free grouping

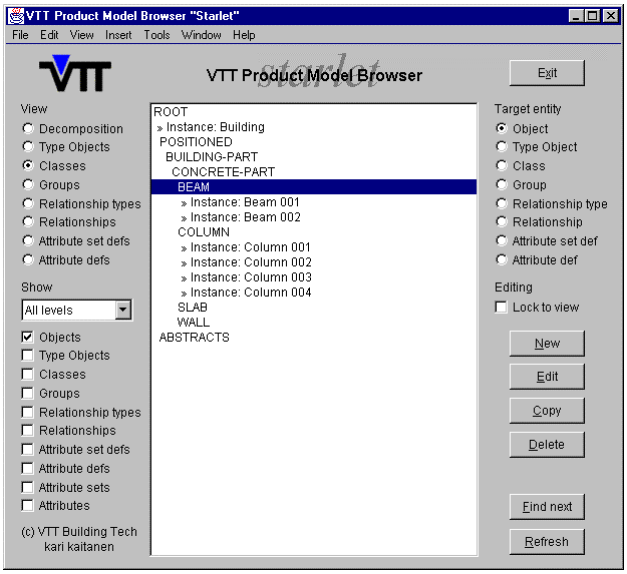
Another very interesting property of the OOCAD 6.0 is that objects can be set to any number and any kind of groups. In any field - but maybe in the construction business especially - it's necessary to offer grouping for different kind of objects. The group can be related to almost anything: ownerships, materials, suppliers, timing, etc. For these reasons the object grouping is on purpose modelled to be very flexible in OOCAD 6.0; for a group any number of subgroups as well as many supergroups are supported. I.e. groups are stored in a 'directed net' for maximum flexibility and minimum redundancy of stored data.

3. "STARLET" - GENERIC PRODUCT MODEL BROWSER

"Starlet" generic product model browser has been implemented with Java. The browser can be opened by any standard 32-bit browser like Netscape or MS Explorer (http://www.vtt.fi/cic/java/starlet/starlet.html) or it can be run as stand-alone program on any environment where Java Virtual Machine is present.

3.1 Main approach

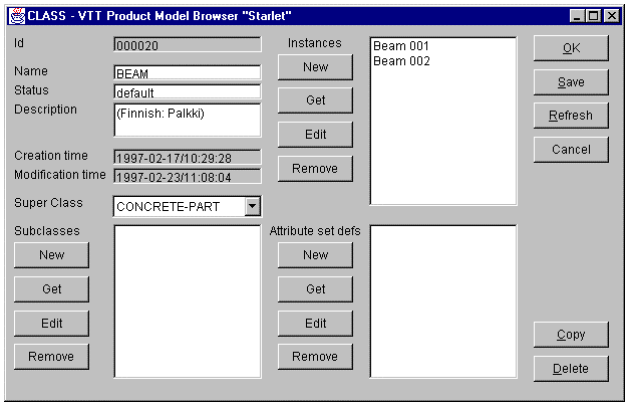
The basic idea of this generic product model browser is that you can examine the product model from many (8) different viewpoints - via Internet. In a way you can understand these different views as 'sorting' the product model based on different entities.



Picture 3.1. Browser's main interface.

In 'Starlet' the product model, i.e. it's objects, groups, classes, etc. can be viewed for example as a decomposition view to the product model; the parent and child objects ('parts of') are shown by intending the child objects. As well you can check the class hierarchy (*picture 3.1*) or examine the objects sorted by groups and their subgroups. Any object can be also be set to exist as a 'type object' and gathered up to the same database, but handled as a separate library.

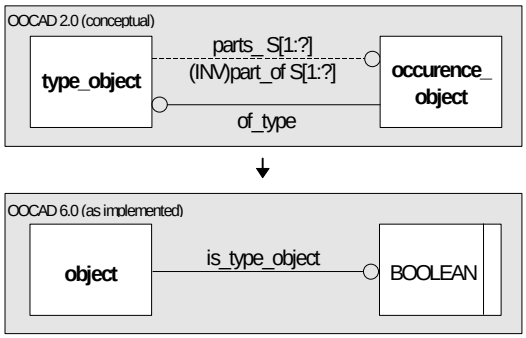
3.2 Dynamic class handling



Picture 3.2. Dynamic class editing.

Classes are also - by the meta-level schema - also generic entities and inherit the same basic properties as all the other entities in schema. Therefore new classes can be created - or destroyed - dynamically in the browser environment (*picture 3.2*). The generic properties: unique entity id, name, status, description and creation as well as the modification times take care that class can be stored into the database as any other entity. New classes are accepted for the database if only the superclass is defined; only one class, defined as rootclass, can be left without the ancestor class.

3.3 Type object vs occurrence objects

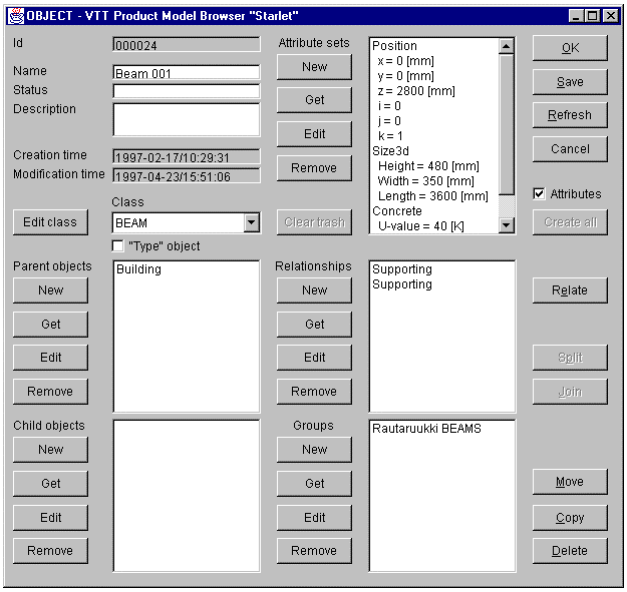


Picture 3.3.
Merging type and occurrence objects.

In generally it's often important to be able to separate logical type objects from occurrence objects. This is sensible not only for saving space in the product model but for keeping the data model as logic as possible. Type objects are objects that are used in the product model in many places; whenever user might think that this particular object might be used also for describing some similar substructure of some other object.

In it's earlier versions [Hannus et al. 1995a/b] the OOCAD schema separated distinctly the type and occurrence object (*picture 3.3*). The problem in these separate objects was that even though it leads to very compact structure of the database it also leads to very inflexible decomposition hierarchy; every other object in the decomposition structure had to be presented as a type object - even if it would surely occur just once in the model.

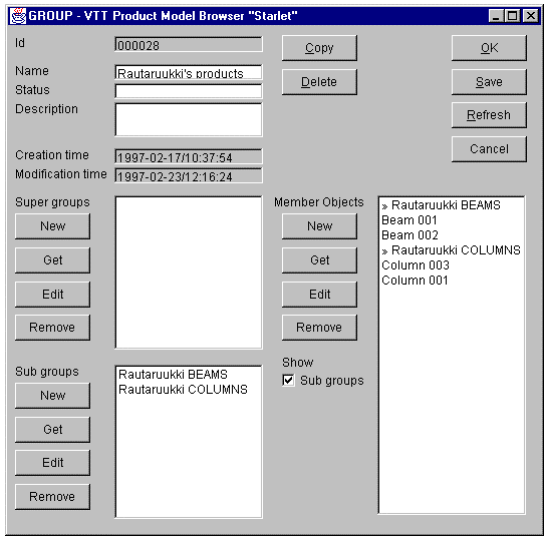
Therefore in the last version of OOCAD schema these type and occurrence objects were merged together and the user may change the status of any object to be a type or an occurrence object (*picture 3.3/3.4*). Though in case the object has more parent objects than one, it's automatically forced to exist as a type object - as it's already presents a logical type object in the product model.



Picture 3.4. Interface of a general object.

3.4 Grouping of objects

Objects can be grouped to any number of groups. Furthermore groups can be set into any number supergroups - and they can have any number of subgroups. I.e. the hierarchy of grouping has not been limited in any way (*picture 3.5*).



Picture 3.5. Object grouping in "Starlet"

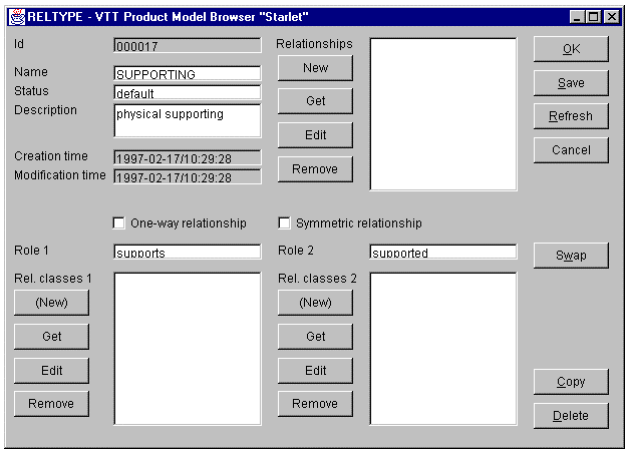
If an object is a member of subgroup it's automatically a member of its' supergroups and all the higher parent groups. In some cases the same object - or group - can also appear in many levels of the same hierarchy-tree; the object could be a member of some group but also a member of its' subgroup. This way for the user has been given the possibility of deleting the object from some place of the grouping hierarchy so that it still might appear in the group in some other place. This characteristics of very flexible object grouping has also taken account in "Starlet" product model browser; the object can appear in many different levels in the group hierarchy tree, but is shown only once to the user.

3.5 Dynamic relationship typing

There are a few common relationship types - as supporting or attaching - that can be found usually in any product model describing the data of buildings. OOCAD 6.0 supports making any amount of relationship types and the user can also create new types dynamically while building the product model (*picture 3.6*).

3.5.1 Symmetric vs asymmetric relationships

Compared to the earlier versions of OO-CAD [Hannus et al. 1995] it's now also possible to categorise some relationship types to 'symmetric' or 'one-way' -relationship types. Symmetric relationships present the relationships that have the same characteristics both ways. For example relationship types 'attaching' or 'touching' are clearly symmetric relationships, as the roles of participating objects are exactly the same both ways. Both objects are 'touching' each other - in case no other specific role is defined - i.e. 'touches' and 'being touched' has exactly the same effect by the participating objects.



Picture 3.6. Creating new relationship types

3.5.2 Objects' different roles in relationships

On the other hand 'supporting' is a good example of asymmetric relationship as the other object has a distinctly different role, e.g. in this case the other object 'supports' when the other is 'supported'. With "Starlet" browser the user can handle both symmetric and asymmetric relationship types (*picture 3.6*). Different relationships can also be limited - by it's role - between objects of some specific earlier determined classes. For example only the objects generated from the 'Beam' -class could be set to share relationships that are 'supporting' other objects. On the other hand some secondary building parts - like windows or doors - could be set only to be 'supported'.

3.6 Attribute handling

By OOCAD 6.0 schema all the attributes are handled in special groups, attribute sets. Attribute sets are defined by an attribute set definition -entity, which stores the information of the attribute list. Attribute definition defines unit and datatype for each attribute. In "Starlet" -browser the attribute definition entities are extended to contain also the default, minimum and maximum values of the attribute [Kaitanen 1995].

"Starlet" -browser has a very flexible way of pointing the attributes so that no redundancy in information storing is needed. One attribute can be pointed from any number of hosts, i.e. attribute sets. That means that for example the same attribute for wall width can be linked to present any other measurement in the building. This characteristics of free pointing can be used splendidly also for printing and viewing the product in 3D or 2D geometry; for each building object there has to be a specific viewing object for this dimension defined - that is in it's simplest just a bounding box. This object can be set to follow exactly to the same parameters of some critical measurements of any 'real' object in the database.

On the other hand this characteristics of pointing to any new or old attribute can be used in almost any other data storing entity in the product model e.g. attribute set, attribute definition or attribute set definition -entities. In principally this gives the user two totally different ways of describing similar objects (copies) to product model: by creating a type object and multiplying its' parent occurrence objects or by pointing to only one attribute set from any number of occurrence objects.

For example a beam of similar measurements can be described in the model - instead of using a specific type object - by using the same attribute set for width, height and length but different attribute sets for position and other data. This wide flexibility of defining similar structures in different ways gives the researcher new good possibilities to study methods of structuring non-redundant product models in practice.

3.7 Database connectivity

Because no outstandingly promising Java supporting object database was found for this project while starting this project, "Starlet" browser is still transferring data to and from a file where the product model is stored. But even though a clear interface between application and (still virtual) database was made, so that in the future it's possible to start using any commercial object database. Please note that now there are already a few very promising object oriented databases available.

3.8 Support to standards (IFC/STEP)

The application doesn't directly support any standard (ISO/STEP or IFC) but as the used schema has been left to be very open - i.e. high level - any standard can be quite easily mapped to OOCAD 6.0. STEP physical files (SPF) will be supported in the future versions for these various (IFC-) schemata (*picture 3.7*).

Please note that any schema of for example IFC standard can be already built inside the browser so that when starting for example a new project (*picture 3.6*) the current used class hierarchy is set as soon you open a new building project based on for example core classes of IFC.

Picture 3.7. SPF I/O

4. CONCLUSION

This paper introduced a new practical approach to product model research by offering a tool - Java based product model browser - for viewing, editing and studying the structures of product models and possible data models in the construction field. The "Starlet" tool was not developed for handling construction projects in practice but is a handy tool for studying possible solutions in similar future applications and data modelling projects.

On the other hand this research does enlighten a lot the excellent possibilities of the new Java programming language in the field of object oriented and nowadays much net-based IT technology. Java gives an excellent ground for new product modelling research by its' robust, multi-threaded architecture fully integrated to Internet.

REFERENCES

[Hannus et al. 1994] Hannus, M., Karstila, K. & Tarandi, **Requirements on standardised building product data models**, In: Scherer, R.J. (ed.) Product and Process Modelling in the Building Industry, Proceedings of ECPPM '94, Dresden 5-7 October 1994. Rotterdam: A.A. Balkema. P. 43 - 50. ISBN 90-5410-584-8

[Hannus et al. 1995a] Hannus, M., Karstila, K.J. & Serén, **Generic product data model for product data exchange - Requirements, model and implementation**, In: Pahl, P.J. & Werner, H. (eds.) Computing in Civil and Building Engineering. Sixth International Conference on Computing in Civil and Building Engineering VI-ICCCBE, Berlin 12-15 July 1995. Vol. 1. Rotterdam: A.A. Balkema. P. 283 - 290. ISBN 90-5410-557-7

[Hannus et al. 1995b] Hannus Matti, Karstila Kari, Serén Karl-Johan, **A generic product data model for building product models**, Espoo, (draft 2.6.95) 1995, VTT.

[Heller et al. 1997] Heller Philip, Roberts Simon, Seymor Peter, McGinn Tom, **Java 1.1, Developer's Handbook**, USA, 1997, 1174 p., SYBEX Inc.; 1460, ISBN 0-7821-1919-0

[Kaitanen 1995] Kaitanen Kari, **Geneerisen tuotetietomallin soveltaminen rakennesuunnitteluun (Applying a generic product data model in structural design)**, Espoo, 1995, VTT, 160 p.

[Serén et al. 1993] Serén, Karl-Johan, etc., **Object-oriented CAD tool implementations for the construction industry**, Espoo, 1993, 90 p., VTT, VTT tiedotteita; 1460, ISBN 951-38-4354-8, ISSN 1235-0605