

CONCEPTUAL MODELLING OF STRUCTURE AND SHAPE OF COMPLEX CIVIL ENGINEERING PROJECTS

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College voor Promoties aangewezen,
op dinsdag 22 september 1998 te 16:00 uur

door

Peter Henri WILLEMS

civiel ingenieur
geboren te Den Haag

Dit proefschrift is goedgekeurd door de promotor:

Prof. ir. F.P. Tolman

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. ir. F.P. Tolman,	Technische Universiteit Delft, promotor
Prof. ir. H.W. Bennenk,	Technische Universiteit Eindhoven
Prof. dr. ir. F.W. Jansen,	Technische Universiteit Delft
Prof. dr. ir. P. van der Veer,	Technische Universiteit Delft
Prof. ir. J. Witteveen,	Technische Universiteit Delft
Dr. ir. G.T. Luiten,	Hollandsche Beton Groep

The research presented in this thesis is sponsored by the Technology Foundation (STW) and TNO Building and Construction Research.

Published and distributed by:

Peter Willems
Dokter Keukenmeesterstraat 4
2265 BK Leidschendam
The Netherlands
Phone: +31 (0)70 3176225
E-mail: p.willems@bouw.tno.nl

ISBN 90-9011888-8

The cover design by Marcel Boender is composed of computer graphics rendered by the ViaDesign package (courtesy of Rijkswaterstaat) and a photograph made by the author.

© Copyright 1998 by Peter Willems

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior permission of the author.

Stellingen

bij het proefschrift

***Conceptual Modelling of
Structure and Shape of Complex
Civil Engineering Projects***

Peter Willems

22 september 1998

1. Een noodzakelijke voorwaarde voor hergebruik van eerder ontworpen en gerealiseerde (onderdelen van) bouwwerken is een geïntegreerde modulaire representatie van productstructuur en vorm.
2. Invoering van PDT (Product Data Technology) zonder aanpassing van het productieproces zal als regel niet tot efficiëntie- en/of kwaliteitsverbetering leiden.
3. Naast een nauwkeurige beschrijving van een bepaald onderdeel van het ontwerp is ook het vastleggen waarom voor deze oplossing is gekozen essentieel voor een effectieve communicatie.
4. Het vastleggen van prestatie-eisen door de opdrachtgever enerzijds en het omschrijven van prestatie-karakteristieken door de opdrachtnemer anderzijds is een zuivere specificatie voor de gewenste informatie-uitwisseling tussen hiërarchisch gescheiden partners.
5. Een modulaire representatie vormt een belangrijke voorwaarde voor de realisering van een over de bouwpartners gedistribueerde up-to-date (evergreen) productmodel.
6. De in de praktijk toegepaste variëteit in het ontwerp van veel onderdelen van civieltechnische werken is zodanig dat de betreffende bouwobjecten geschikt zijn om door een parametrisch model te worden beschreven.
7. Toepassing van PDT in GIS (Geografisch Informatie Systeem) applicaties kan de huidige functionaliteit (presenteren van gegevens met een geografische component) op een niveau brengen waarop betekenisvol met CAD-pakketten kan worden gecommuniceerd.
8. Omdat de informatie die in DNA is vastgelegd meer het karakter van een kookboek dan van een blauwdruk heeft, zou men parametrisch modelleren een 'natuurlijke' wijze van representeren kunnen noemen.

9. Onze hersenen en de rest van de wereld zijn alleen met elkaar verbonden door overdracht van informatie (input via de zintuigen, output via de spraak of aansturing van het lichaam), men kan dus stellen: ik wissel informatie uit dus ik besta.
10. De verhoogde efficiëntie die met de ondersteuning van computer-applicaties kan worden bereikt wordt veelvuldig weer teniet gedaan door de snelle opeenvolging van de versies, die vaak onderling incompatibel zijn.
11. Borden met het opschrift "Verkeerssituatie gewijzigd" dienen van een ingangsdatum te worden voorzien.
12. De millennium-crisis is eigenlijk een centennium-crisis.

Table of Contents

List of Abbreviations.....	ix
1 Preface	1
2 The Problem.....	5
2.1 Introduction	5
2.2 The Construction Industry Is Different	5
2.2.1 The Product Domain.....	6
2.2.2 The Realisation Process.....	7
2.2.3 The Co-operation Structure	9
2.3 Conclusions	10
2.4 Research Questions	11
3 Representation of Structure and Shape	13
3.1 Definition, Representation and Presentation	13
3.2 Shape as the Carrier for Product Definition Data.....	21
3.2.1 Drawings.....	22
3.2.2 Three Dimensional Models	23
3.3 Product Modelling	26
3.4 Conclusions	26
3.5 Research Questions.....	28
4 Product Modelling Structures.....	31
4.1 Introduction	31
4.2 Objectives for product data structures	35
4.3 Product, process or peripheral	36
4.3.1 The association between products and processes	36
4.3.2 The association between in-scope and out-of-scope concepts	37
4.3.3 Shape aspects of products, processes and peripherals	38
4.4 Specification	38
4.4.1 Generic, type and occurrence specification levels	38
4.4.2 Shape aspects of specification levels	40
4.5 Decomposition	41
4.5.1 Decomposition structures	41
4.5.2 Modularity	44
4.5.3 Concretisation	45
4.5.4 Shape aspects of decomposition.....	47

4.6 Connectivity	47
4.6.1 Connectivity network specification	48
4.6.2 Modularity	49
4.6.3 Properties	53
4.6.4 Shape aspects of connectivity	54
4.7 Life cycle.....	55
4.8 A Product and Process reference model	57
4.8.1 Model development in conformance with a reference model	58
4.8.2 Product type models	60
4.9 Conclusions	62
4.10 Research Questions	63
5 Shape Modelling	65
5.1 Introduction	65
5.2 Space Modelling	66
5.2.1 Dimensionality.....	66
5.2.2 Embedding	68
5.2.3 Symbolic notation.....	71
5.2.4 Space graph composition.....	73
5.2.5 Spaces data model	77
5.3 Cell Modelling	77
5.3.1 Dimensionality.....	77
5.3.2 Symbolic notation.....	78
5.3.3 Enclosing and Bounding	79
5.3.4 The interface between cells and spaces.....	82
5.3.5 Inclusion and exclusion	84
5.3.6 Cell graph composition	89
5.3.7 Cells data model	90
5.4 Shape Modelling	91
5.4.1 Shape Primitive.....	91
5.4.2 Axiomatic shape primitive set.....	92
5.4.3 Derived shape primitive set.....	93
5.4.4 Shape Complex	94
5.4.5 Shape Occurrence	97
5.4.6 Shapes data model	99
5.5 Conclusions	100
5.6 Research Questions	100
6 Integration of Product Modelling and Shape Modelling	103
6.1 Idealisation and accuracy	104
6.1.1 Maximum and minimum shape	107

6.1.2 Idealised shape.....	108
6.1.3 Shapes of different dimensionality.....	109
6.1.4 Required shape	110
6.2 Shape specification	110
6.3 Decomposition	113
6.4 Connectivity: Port shape	115
6.5 Port decomposition	121
6.6 Conclusions	122
7 Case 1: Roads.....	125
7.1 Introduction	125
7.1.1 Historical overview	125
7.1.2 Modelling issues	127
7.2 Road Shape Model Kernel	128
7.2.1 Chains	129
7.2.2 Horizontal and vertical alignment	130
7.2.3 Road	133
7.2.4 Super elevation and width variations	137
7.3 Shape evaluation	140
7.3.1 Shape evaluation steps.....	141
7.3.2 Assemblage of the shape graph	142
7.3.3 Assemblage of complex deformation components	146
7.4 Conclusions	147
8 Case 2: Viaducts	149
8.1 Introduction	149
8.2 Viaduct support structure definition	150
8.2.1 The relationship between road design and viaduct design	150
8.2.2 Viaduct support structure model features	151
8.2.3 Shape evaluation	154
8.3 Parametric viaduct modeller	155
8.3.1 The ESPRIT GEM project and the Fly-over demo	155
8.3.2 Viaduct parametrics	155
8.3.3 Shape evaluation and B-rep generation	158
8.4 Conclusions	160
9 Summary, Conclusions and Recommendations	161
9.1 Summary	161
9.1.1 The problem.....	161
9.1.2 The proposed solution	162
9.1.3 Evaluation of the case studies	165
9.2 Conclusions	167

9.2.1 Research question 1	167
9.2.2 Research question 2	168
9.2.3 Research question 3	169
9.3 Recommendations	170
Samenvatting	173
1 Het probleem	173
2 De voorgestelde oplossing	174
3 Evaluatie van de praktijkvoorbeelden	177
References	181
Road Shape Model Kernel.....	187
Curriculum Vitae	205
Index	207

List of Abbreviations

AEC	Architecture, Engineering and Construction
AIC	Application Interpreted Construct
ANSI	American National Standards Institute
AP	Application Protocol
API	Application Programming Interface
ATLAS	Architecture, methodology and Tools for computer-integrated LArge-Scale engineering, ESPRIT project 7280
B-rep	Boundary representation
BCCM	Building Construction Core Model
CAD	Computer Aided Design
CAX	Computer Aided system
CIC	Computer Integrated Construction
CIM	Computer Integrated Manufacturing
CIM-OSA	Open System Architecture for CIM, ESPRIT project 688
CIME	Computer Integrated Manufacturing and Engineering
COMBINE	Computer Models for the Building INdustry in Europe, JOULE project
CORBA	Common Object Request Broker Architecture
CSG	Constructive Solid Geometry
DAG	Directed A-cyclic Graph
DIANA	DIplacement ANALysis
DTM	Digital Terrain Model
DXF	Drawing eXchange File format (AutoCAD)
EPISTLE	European Process Industries STEP Technical Liaison Executive
ESPRIT	European Strategic Programme for Research and Development of Information Technology
FEM	Finite Element Method
GARM	General AEC Reference Model
GEM	Generic Engineering analysis Model, ESPRIT project 8894
GKS	Graphical Kernel Standard
HTML	Hyper Text Mark-up Language
HVAC	Heating, Ventilation and Air-Conditioning
IAI	International Alliance for Interoperability
IDEF	Integrated DEFinition
IDM	Integrated Design Model
IFC	Industry Foundation Classes

IGES	Initial Graphics Exchange Specification
IMPACT	Integrated Modelling of Products and Processes using Advanced Computer Technologies, ESPRIT project 2165
ISO	International Organisation for Standardisation
IT	Information Technology
LPM	Logical Product Model
LSE	Large Scale Engineering
MARITIME	Modelling And Reuse of Information over TIME, ESPRIT project 6041
MOSS	Modelling SystemS
NC	Numerically Controlled
NIAM	Nijssen's Information Analysis Methodology
NIDDESC	Navy-Industry Digital Data Exchange Standards Committee
NIST	National Institute for Standards and Technology
OpenGL	Open Graphics Library
P&P model	Product and Process model
PDES	Product Data Exchange using STEP
PDT	Product Data Technology
PISA	Platform for Information Sharing by CIME Applications, ESPRIT project 6876
PMshell	Project/Product/Process Modelling shell
PoC	Proof of Concept
POSC	Petrotechnical Open Software Corporation
ProMod	Product Modeler
RMI	Remote Method Invocation
RMK	Road Model Kernel
RSMK	Road Shape Model Kernel
SDAI	Standard Data Access Interface
SDT	Space Deformation Tree
SQL	Structured Query Language
STEP	STandard for the Exchange of Product model data
STW	Stichting Technische Wetenschappen (Technology Foundation)
TILLY	Transient and static analysis, Incremental loading, Linear and non-linear behaviour, Lumped masses, springs and dash-pots, Young and ageing material
TNO	Netherlands Organisation for applied Scientific Research
UoF	Unit of Functionality
ViaDesign	Viaduct Design System
VRML	Virtual Reality Modeling Language

1 Preface

The introduction of computers to support the product design and manufacturing processes affected initially only the execution of certain tasks like drawing, planning or engineering calculations. Until recently the information streams between these automated or computer aided activities, the communication with project partners and archiving were mostly paper-based. This situation has become more and more a bottleneck for further improvement of the product development process, i.e.

- the reduction of the time-to-market of a product,
- the reduction of product development costs and
- the increase of the quality of the product.

Bringing in the information flows into the area that is controlled by the automated information system seems to be the obvious solution. In other words computers should not only support certain business processes, but also handle the communication between these processes.

Automated information sharing or exchange is only feasible if both sides (sender and receiver) agree upon the applied protocol or how to represent the exchanged data. The various digital representation forms of product information betray often the discipline that originally has introduced that particular representation. Over the years the computer aided drawing and design development has brought forth the area now known as geometric modelling, i.e. representing shape definition in a digital format. However, representation forms for geometric modelling are less equipped to express things like design intent or constraints. This gave way to other representation forms as feature modelling, parametric modelling or more in general product modelling.

The aeroplane and automotive industry are the driving forces behind these developments. The different characteristics of the construction industry have prevented a straight employment (without adaptations) of the technical solutions and tools that were originally developed by the mechanical industry. An outcome of this thesis is a new modelling approach for civil engineering works like roads, railroads, tunnels, viaducts and their combinations.

The main characteristics of this new approach are that it supports:

- rapid modelling of individual shapes,
- re-uses of earlier designs and
- interfacing to knowledge-based applications.

Product structure (composition, connectivity) and product shape are the two modelling areas that establish the foundation, as well a proposal how to integrate these modelling areas more firmly.

The structure of this thesis is as follows:

- Chapter 2 *The Problem*
introduces the problem and defines a first formulation of the research goal.
- Chapter 3 *Representation of Structure and Shape*
analyses the state of the art of digital shape representation and product definition.
- Chapter 4 *Product Modelling Structures*
discusses various basic constructs to organise product data in an information model.
- Chapter 5 *Shape Modelling*
discusses a new representation for structured shape modelling.
- Chapter 6 *Integration of Product modelling and Shape modelling*
discusses the options to integrate product modelling and shape modelling structures of the previous chapters.
- Chapter 7 *Case 1: Roads*
describes a case study derived from a project to develop a highway information model.
- Chapter 8 *Case 2: Viaducts*
describes a case study derived from two viaduct information model developments.
- Chapter 9 *Summary, Conclusions and Recommendations*

This thesis is the final conclusion of a PhD research that I started almost eight years ago. During that period I have gratefully made use of appropriate TNO projects that I was involved in. Initially they introduced me to the problem field of product information exchange and later provided an environment to test many of the developed concepts discussed in this thesis.

Research is always a team effort. Without the inspiring guidance of my supervisor, Frits Tolman, and the stimulating climate provided by my colleagues

(former and present) at TNO, this PhD research would simply be inconceivable. I would like to mention here a few names. Wim Gielingh, who invented many basic concepts I have used as a foundation for my own contributions. Peter Kuiper, who motivated me to undertake this effort and managed the financial and organisational framework to make it feasible. My fellow PhD students of the STW/CIC (Computer Integrated Construction) project, Bart Luiten, Wim Bakkeren, Ronald Krom and Sander van Nederveen, always willing to think along, find weak spots, test prototype software, correct manuscripts, and so on. Bart Luijten, who always responded very rapidly if I needed new features in the product modelling tool PMshell. Frank Mol who, as project manager of Rijkswaterstaat, meticulously analysed all modelling features in the various highway information models I developed.

Finally, I would like to thank my family, Tilly, Tessa and Vicky, for all the patience and support during these years.

Leidschendam, August 1998

Peter Willems

2

The Problem

Why is the construction industry lagging behind in the application of information technology? What makes the construction industry different when compared with other branches of industry? How have these peculiarities affected the way information about the products and the manufacturing processes is recorded, communicated and archived? Moreover, how can the application of computers and digital media additionally improve the efficiency of the production process and the quality of its products?

2.1 Introduction

The subject of this thesis is the electronic description (or representation) of structure (topology) and shape (geometry) in the construction industry. The presented concepts appear particularly advantageous for civil engineering works, specifically of linear infra-structure works like roads and railways, with crossings, bridges, viaducts, tunnels and so on.

This chapter introduces the problems with the current state-of-the-art of information technology (IT) usage in the construction industry, which can be clarified by several unique characteristics of this branch of industry.

2.2 The Construction Industry Is Different

Compared with other branches of industry, particularly the mechanical industry, the construction industry appears to be hesitant and less successful in the employment of computers in the primary business process chain (design and manufacturing processes). Which aspects of the idiosyncrasies of the construction industry may clarify this contrast in accomplishment? Of course, there is no general answer to this question because of the internal differences between the various sub-branches of the construction industry and between each individual company. However, to present a general impression of those aspects,

that probably influence the referred to distinction in successful employment of information technology, the specific characteristics of the mechanical industry and the construction industry are compared. This comparison will be structured along three different topics:

- the product domain (what)
what kind of things are manufactured and what are the typical characteristics of these products,
- the realisation process (where, how and when),
which conditions determine the realisation of the manufacturing process,
- the co-operation structure of the involved partners or sub-contractors (who),
how are the different roles (initiator or commissioner, designer, (sub-)contractor) in the realisation process distributed? Finally, how do these actors communicate (exchange information) with each other?

2.2.1 The Product Domain

Civil engineering works are often large and complex (buildings like offices, hospitals, airport terminals) to very large and very complex (interrelated infrastructure objects like highways, railways, bridges; or water-works like harbours, dikes, dams). They have typical dimension magnitudes that range from 10^{+1} m to 10^{+4} m, and they have a one-to-one relationship with a fixed geographical location¹. Mechanical products show a large variation in sizes (roughly between 10^{-3} m and 10^{+2} m), besides a mechanical product is typically mobile, i.e. it is not permanently attached to a geographical site. This close relationship between a building object and its geographical environment makes it sometimes hard to decide where to draw the line between product and environment, e.g. between a road and the landscape.

Besides this integration of civil engineering products into their environment, the integration among civil engineering products is an important factor for increased complexity. Viaducts and tunnels, for instance, must conform to the same overall highway alignments. This applies both for the newly designed road section as for the already existing infra-structure.

Civil engineering works are produced in very small series (often just one instance), while the occurrences in a series are seldom really identical. On the other hand, mechanical products are rarely manufactured as one-of-a-kind products². Obviously, the dominant one-of-a-kind character of the product

¹ Exceptions are off-shore structures like drilling platforms.

² Although e.g. a car can be manufactured according to various configuration parameters, which may technically lead to millions of different product versions, each car is basically still an instance of a certain car type.

domain of the construction industry spreads its influence over virtually any aspect of this branch of industry. The total price of a singular end product will directly include its overall costs of design. Such a direct financial link is a natural barrier to achieve a high degree of optimisation. With the result that design alternatives are analysed in a very early stage to select a single thread, i.e. proto-typing is practically unknown in the construction industry³.

The variation in shape is probably the most unmistakable characteristic of the one-of-a-kind essence of civil engineering works. This high degree of variation in shape and the one-of-a-kind character of civil engineering works are apparently tightly interrelated and appreciated as tailor-made solutions to meet each specific requirement. The consequences are that:

- *earlier realised designs are seldom re-used.*

It turns out that, although the benefits of re-using a previous realised design are obvious, each new project appears to have a unique set of requirements that demands a unique solution.

- *symmetry in shape features is low.*

The overall shape frequently suggests symmetry (mirrored symmetry or translated symmetry), however this is seldom the case. A one-of-a-kind production commands insufficiently the advantages of a genuine symmetrical shape.

- *geometry is virtually always unique.*

Although building shapes may be the same in a topological sense (i.e. same number and orientation of faces, edges, corners) in a geometrical sense (locations, gradients) they appear to be unique. Notwithstanding the monotony of a highway from a driver's perspective, each section has a different geometric alignment.

This high degree of variation in shape opposes the observation that the structure (or topology) of instances of the same civil engineering works product family (for example all viaducts) do not show that much variation. Splitting structure and shape could be a conceivable course to avoid the 'starting from scratch' approach, which is the dominant practice in the construction industry.

2.2.2 The Realisation Process

The typology of the product domain (large to very large object dimensions, close integration with its geographical environment and mostly one-of-a-kind) imposes severe restrictions to the manufacturing process. For obvious reasons the location for the operational life stage of the building object and the location

³ Of course, one could also state that the construction industry produces *only* proto-types.

for manufacturing it must be virtually the same. As a result the 'factory' for manufacturing a building object can be considered as a throw-away facility to materialise a single product instance. Although the components of this facility are mostly reusable, its configuration is principally improvised. Each new building project must first assemble this temporary facility before the actual manufacturing process can start. This approach in particular makes the construction industry very different even compared with neighbouring branches of industry like shipbuilding⁴. It has traditionally favoured contractors with improvisation skills, however, at the same time it has obstructed the deployment of dedicated autonomous machines (industrial robots) in the production process. These devices demand, at least for the moment, a much more stable environment.

Besides the location of the manufacturing process the process itself has also 'different' properties. The relative uniqueness of building objects has influenced the repetition in manufacturing steps possibilities in a negative sense, which in its turn is an obstacle for an efficient use of machines. In addition the level of standardisation (standard dimensions, standard parts, standard details) is, compared with other branches of industry, low. As a result the machines that are used successfully must be widely applicable and are better characterised as manufacturing tools rather than manufacturing machines. A manufacturing machine has here the meaning of a device that realises (semi-) autonomously a particular step or a series of steps in the manufacturing process [Krom 1997].

The timing aspect of the manufacturing process has also some distinguished properties. Time pressures are often extremely high, for example it is not uncommon to start the manufacturing process before the completion of the design. Furthermore, a large number of circumstances may have a negative influence. These are for instance:

- a strong dependency from weather conditions,
- a significant influence of the conditions of the soil,
- substantial delays may be imposed by democratic decision procedures,
- a site may have an insufficient storage capacity, or
- a site can be difficult to access for the delivery of (large) building materials.

Minor influences are:

- the security issues of fencing off the building site against theft and vandalism, or by
- the discovery of archaeological artefacts.

⁴ If a shipbuilder would choose for a similar manufacturing approach he would build a ship at the site of the commissioner instead of his yard.

Finally, a set of influences, which are not typical for the construction industry but nevertheless rapidly alter the traditional routine practices of the manufacturing process:

- reduced production times
- late modifications in required characteristics
- increase of regulations
- increase of competition
- increase of quality level

2.2.3 The Co-operation Structure

An important aspect in the realisation of a building object is the fact that the design process and the materialisation process are traditionally not carried out by one and the same company. This means that the milestone ‘design finalised’ and ‘start of materialisation’ coincides with a change of company and, as a result, a very explicit information exchange of the design (formal specification).⁵ This fact is again quite unique for the construction industry. In comparison a shipbuilder keeps the two major manufacturing processes in one hand.

The many relatively independently operating sub-contractors (in contemporary terms constituting a virtual enterprise), that realise the various building systems (structural system, electrical system, HVAC systems, and so on), are also typical for the co-operation structure. This co-operation structure is only feasible with either a superb information management system (which is traditionally not available) or, again, a high degree of improvisation skills of the participating sub-contractors.

The following circumstances are important obstacles to come to an agreement about the procedure how to exchange information:

- the continuously changing co-operation alliances between companies,
- the introduction of new requirements, building codes and regulations, and
- time pressures, which lead to concurrent design and engineering, which seems to make the information exchange even more complex than in a sequentially oriented realisation process.

However, the quality of the information system could be the key factor to bring the construction industry on a higher level of efficiency, facilitating:

⁵ There is a trend towards so-called Design/Construct contracts keeping the total realisation process in one hand. However, even then deadline pressure forces often to commence the manufacturing process before the finalisation of the design process, disturbing an efficient and optimised construction process execution.

- a more optimised design by raising the reusability of the design of particular building components,
- a higher deployment of dedicated autonomous machines in the manufacturing process,
- a more smooth transition from design stage to manufacturing stage,
- a better integration of sub-contractor activities.

To realise these goals it is necessary that (without substantial additional costs) the designs can be described in detail by computers and that these electronic descriptions can be communicated with the CAX applications of all the participants in a building project. This scenario can be feasible if it leads to a drastic reduction of the costs that are spent now to perform the inter-partner communication in a traditional manner.

2.3 Conclusions

The one-of-a-kind character of its products and the site-bounded, highly flexible but temporary characteristics of its organisation and production process are probably the critical factors that explain the hesitant admission of information technology in the construction industry.

Civil engineering works often show complex interrelations both in structure (topology) and shape (geometry). Particularly the shape characteristics of instances of the same product family show a substantial level of variation. As a result re-use of earlier designs is the exception while 'starting from scratch' seems to be the common practice.

Compared with other branches of industry the construction industry stays well behind in employing contemporary information technology in the realisation of a project. Although nowadays computers support many activities, the communication infra-structure in a building project is still inadequate to harvest the true benefits of a computerised information system. Low level information interfaces are short-lived under these conditions because they lack the necessary flexibility⁶. A high quality communication infra-structure is in principle capable to raise the level of efficiency of the production process and the quality of its products. A higher level of meaning or semantics seems to be the key factor

⁶ A direct translator, for example, between two applications must be adapted for each new release of one of these applications. The order of adaptations is therefore squared with the order of new releases.

for improving the communication between computers and between computers and humans in the construction industry.

Within the total ‘universe of discourse’ of the construction industry this thesis will utilise projects, that were carried out in the same time span (1991-1997), to zoom-in on the sub-domain of complex topologies and shape definition and representation of civil engineering works, especially linear infra-structural works. It will investigate the possibilities to bring it on a more meaningful level than ‘just a set of lines and surfaces’. This means that the intersection of the world of geometric modelling and the world of product and process modelling constitutes the focal point.

2.4 Research Questions

From the previous comparison of the construction industry with other branches of industry, the following research questions can be formulated.

First of all there is a need for a general communication protocol for the sharing and exchange of product and process information between the partners in a project, which is stable enough to last for a longer period of time. The related question is:

1. *Which product and process representation is able to meet the requirements of the construction industry?*

This thesis also targets the rapid modelling of individual shapes of civil engineering works, especially linear infra-structural works, supporting the development of parametric linear infra-structural design systems. The research question is:

2. *Which shape representation is able to meet the requirements of the construction industry?*

The last problem researched in this thesis is how to combine the proposed solution for managing complexity, with the proposed shape representation:

3. *How to integrate both (product/process and shape) representations?*

Up to this point the formulation of the research questions is necessarily very general. They will be further refined and supplemented as an introduction for each successive analysis chapter (chapters 3, 4, 5 and 6).

So far, all research questions mention the concept of representation. The next chapter will discuss this concept and its utilisation in the context of this thesis to describe the structure and shape of a building product.

3

Representation of Structure and Shape

To be able to communicate information between two parties (either humans or machines or mixed), a representation must be selected for the messages to be exchanged that suits the chosen communication medium and refers to a semantic level that is understandable by both sender and receiver. Traditionally, the construction industry uses technical drawings as the prime representation of building products. The drawing representation, in principle concentrated on human interpretation, is much less appropriate for state-of-the-art machine interpretation. For this reason computer representation of product shape and structure is an important topic over the last 20 years.

3.1 Definition, Representation and Presentation

In human-to-human communication there are numerous ways to describe a certain object. In a normal conversation, for instance, the sending human could refer to 'my house' if the receiving human has visited this house before or if the type of house is not particularly relevant for the conversation. However, if the sending human wants to sell his house, by means of an advertisement in the local newspaper, the type of house is definitely relevant and he cannot rely that this knowledge is available with the receiving party. In other words he should be more specific: flat, corner-house or villa, so-many storeys, so-many rooms, living-room dimensions, garden dimensions and orientation, and so on. If the sending human wants to inform the mover where to place his furniture, he will probably draw a sketch showing a plan view of his house with directions where to put the various pieces of furniture. Such a sketch needs not to be a very precisely scaled drawing only a rough indication will be sufficient. Finally, if the sending human wants to build a new house the contractor needs a very precise description (specification), both in words and in a graphical format (technical drawing).

The general idea is that the receiving party needs more data if the object of communication is:

- relevant in the scope of the conversation, and/or
- unknown to the receiving party, i.e. the sending party cannot refer to common knowledge, and/or
- necessary for the receiving party to act upon, i.e. the receiving party has to make more or less comprehensive decisions using the object description data.

This situation is not fundamentally different if the receiving party is not a human but a machine (computer). However, with the state of the art in computer science⁷, there is initially no common-sense knowledge available. Common-sense knowledge is common knowledge about the world that is possessed by every schoolchild and the methods for making obvious inferences from this knowledge [Davis 1990]. This common-sense knowledge must be formally defined in conceptual models and procedures that are available both to the sending and the receiving party. For instance, if both parties know the concept of a rectangle the communication of a specific rectangle can be like this:

```
rectangle (length=10.0, width=4.0);
```

However, if the receiving party does not know what a rectangle is it will be incapable to process this statement. One way to overcome such a situation is to try to give a prescription how to build the unknown concept in terms of more general known basic concepts, for example:

```
line (length=10.0); rotate (angle=90.0);  
line (length=4.0); rotate (angle=90.0);  
line (length=10.0); rotate (angle=90.0);  
line (length=4.0);
```

With this recipe the receiving party is able to draw the rectangle, however, the semantics that this is actually a rectangle is lost. If the receiving party does know the concept of a rectangle after all, it is unable to decide if this line configuration was intentional a rectangle or just four lines that by coincidence resemble a rectangle. This situation is typical for the exchange of information between a sending application using a semantically rich representation (high level concepts) and a receiving application using only a set of basic concepts.

⁷ The state of the art in artificial intelligence is regarded here to be still in an academic stage.

Although, the figures presented on the screen may look the same, some basic structuring information is entirely lost. For instance, transforming a circle into a poly-line will make it practically impossible to change its radius.

Another way to deal with an unknown concept is to define it the first time in a conceptual (meta) language, for example:

```
procedure rectangle (real l, real w) {  
  assert ( l > 0.0); assert ( w > 0.0);  
  line (length=l); rotate (angle=90.0);  
  line (length=w); rotate (angle=90.0);  
  line (length=l); rotate (angle=90.0);  
  line (length=w);  
}
```

With this piece of information the receiving party is able to grasp what the concept of a rectangle means, i.e. a figure consisting of four trimmed straight lines connected perpendicularly to each other at their end-points with opposite edges having the same size.⁸ The formal arguments of the rectangle procedure define how the rectangle can be instantiated or transformed, while preserving the rectangle concept.⁹ This conceptual description is part of the *definition* of the concept rectangle. A definition may take many forms, ranging from a precise formal description to an informal textual explanation. A definition should make clear

- how this concept can be distinguished from neighbouring concepts,
- how it fits in a sub/super-type hierarchy (ancestor/descendent concepts),
- which other concepts it refers to (supplier concepts) or
- by which other concepts it is referred by (client concepts) and
- its invariant and variant properties.

Part of this definition may be one or more procedures how to generate a *representation* for an instance of this concept.

To clarify what is meant with ‘representation’ the meaning triangle (figure 3.1) as described in [Sowa 1984] may be helpful. On top is the concept (intension, idea, thought) that refers to the referent (object, extension), something in the real world. A representation fulfils the role of the symbols (words, icons, drawings) that symbolises a concept and stands for an object in the real world.

⁸ The property that it is a closed figure can be derived from this description.

⁹ A very practical reason for this approach of learning new concepts through embedded meta-data could be the reduction of data to be exchanged (if for instance large numbers of rectangles will follow). On a semantically much lower level data compression algorithms use a similar approach.

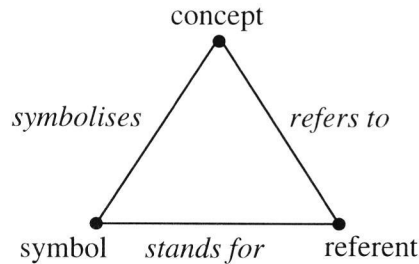


Figure 3.1 The meaning triangle

Representations of a concept instance can be ordered on a scale from implicit to explicit representations. An *implicit* representation is a very compact description that symbolises the concept directly and supposes that the receiving party understands this specific concept and has this particular representation in its repertoire. An *explicit* representation decomposes a complex concept in a set of less complex concepts and uses representations for those constituent concepts. An even more explicit representation could subsequently break down this structure in even more simple concepts. As a result explicit representations will always use more storage or bandwidth (in terms of bits) than implicit representations. Another effect is the loss of structure and semantic content in the direction from implicit to explicit. For this effect it is always possible to design an algorithm that derives a more explicit representation out of a more implicit representation. Vice versa is very hard and often erroneous or simply impossible.

The following enumeration shows a sample of possible representations for the earlier described instance of a rectangle concept:

- *parametric (feature) representation*

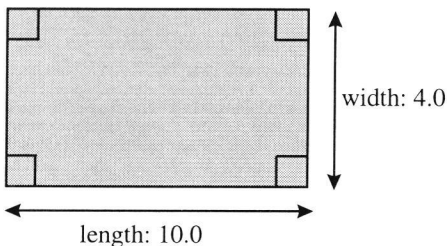


Figure 3.2 Parametric rectangle representation.

```
rectangle(length= 10.0, width= 4.0);
```


A parametric representation is the most implicit representation. Some fine tuning is possible within the sub-super type hierarchy. For example, a rectangle is a descendent of a parallelogram, trapezium, quadrangle, polygon and an ancestor of square. Representing a rectangle as a quadrangle is less implicit and less meaningful, on the other hand it is more meaningful (and more implicit) to call a rectangle with *intentional* equal sides a square. The word intentional is stressed here to make a distinction with accidental congruence.

A parametric representation is based on a set of rules (constraints) to guarantee that each valid set of parameter values always point to exactly one valid object instance. Rules could prescribe here, for instance, the relative locations of the vertices: (x, y) , $(x+\text{length}, y)$, $(x+\text{length}, y+\text{width})$, $(x, y+\text{width})$ or the fixed angles of 90° , or the dimensions of the edges and so on.

This rectangle sample representation does not contain absolute location data. To attach this information a particular fixed point of the rectangle, sometimes referred to as the anchor point (e.g. the top and left corner), should be selected. If the location and orientation of this anchor point are fixed with regard to a co-ordinate system then the location and orientation of each other point of the (rigid) shape can be calculated.

A special kind of parametric representation is Constructive Solid Geometry (CSG) [Mäntylä 1988]. CSG uses a tree-like location and Boolean operation¹⁰ graph structure to assemble complex solid objects from parametric defined primitive solid objects like cube, sphere, cone or torus. The CSG data structure not only stores the parameter values but also a prescription how to build a complex shape (represented by the root object) from a set of primitive shapes (the leave objects).

- *sweep representation*

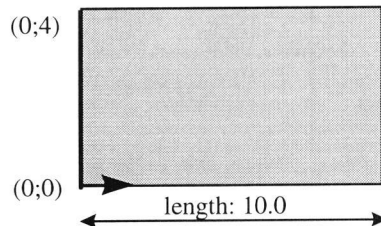


Figure 3.3 Rectangle representation using a sweep operation.

¹⁰ Set operations (like union, difference, intersection) applied to point set operands.

```
#1= line_segment(start= (0.0, 0.0), end= (0.0, 4.0));
#2= direction(x= 1.0, y= 0.0);
#3= path(direction= #2, length= 10.0)
#4= translational_sweep(profile= #1, path= #3);
```

A sweep representation moves a profile shape along a path (translation sweep) or around an axis (rotational sweep) and considers the region the profile shape has traversed as belonging to the result shape. In this case the profile is a straight line segment (length 4.0) aligned with the y-axis and the sweep path a straight line segment (length 10.0) aligned with the x-axis.

Sweep operations can create very complex shapes by using a curved traversal path or by varying the orientation and the shape of the profile as function of the position on the sweep path. Most CSG-modellers have added the sweep-object as a supplementary primitive solid object, which increases the modelling freedom considerably.

- *geometry/topology representation*

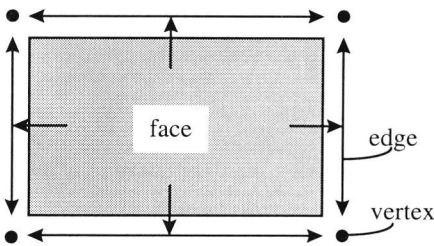


Figure 3.4 Topological rectangle representation.

```
#1= point(x= 0.0, y= 0.0);
#2= point(x= 10.0, y= 0.0);
#3= point(x= 10.0, y= 4.0);
#4= point(x= 0.0, y= 4.0);
#5= line(point= #1, direction= (1.0, 0.0));
#6= line(point= #2, direction= (0.0, 1.0));
#7= line(point= #3, direction= (-1.0, 0.0));
#8= line(point= #4, direction= (0.0, -1.0));
#9= vertex(geometry= #1);
#10= vertex(geometry= #2);
#11= vertex(geometry= #3);
#12= vertex(geometry= #4);
#13= edge(start_vrtx= #9, end_vrtx= #10, geometry= #5);
#14= edge(start_vrtx= #10, end_vrtx= #11, geometry= #6);
#15= edge(start_vrtx= #11, end_vrtx= #12, geometry= #7);
#16= edge(start_vrtx= #12, end_vrtx= #9, geometry= #8);
#17= loop(edge_list= (#13, #14, #15, #16));
#18= face(loop_list= (#17));
```

A geometry/topology representation breaks a shape down into topological

entities (vertices, edges, faces, ...) preserving the connectivity structure (adjacency relations). The nodes of the topological framework generally reference corresponding geometric counterparts (points, curves, surfaces, ...) fixing the *elastic* topological shape into a *rigid* specific shape. With regard to the rectangle example: the concept of a rectangle is lost but its structure of a closed figure with straight edges has remained. Moreover any quadrangular shape will adopt this very same topological framework leaving the differences to the geometric part.

This type of representation is often referred to as *Boundary representation* (B-rep) [Mäntylä 1988], because it determines the shape of an object by stating the contour of the region in Euclidean space that the object occupies (instead of defining the contents directly as for instance CSG does). B-rep is a more explicit representation than CSG because it is (relatively) straightforward to derive a B-rep from a CSG-tree (several CAD-systems have a CSG-based user interface while the modelling kernel itself is based on B-rep), however the reverse transition is in general impossible. Another indication is that B-rep uses much more storage than CSG for an equivalent shape model.

- *vector representation (absolute location)*

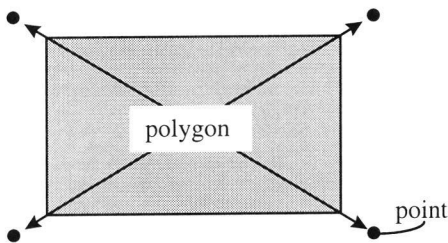


Figure 3.5 Rectangle represented by a polygon.

```
#1= point(x= 0.0, y= 0.0);  
#2= point(x= 10.0, y= 0.0);  
#3= point(x= 10.0, y= 4.0);  
#4= point(x= 0.0, y= 4.0);  
#5= polygon(point_list= (#1, #2, #3, #4));
```

A vector representation supports basically only three shape primitives: point, composite line (poly-line) and n-angular closed figures (polygon). Geometry is represented only by location data, i.e. curves are approximately represented by a set of arbitrary points connected by straight line segments. A vector representation is in general not a suitable representation for the

modelling kernel of a CAD-system. However, this type of representation is very well qualified to support fast rendering of shaded images¹¹.

- *vector representation (relative location)*

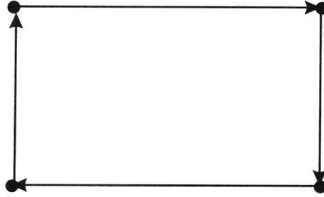


Figure 3.6 Rectangle represented by a poly-line.

```
#1= pen(down= .true.);
#2= move(delta_x= +10.0, delta_y= 0.0);
#3= move(delta_x= 0.0, delta_y= +4.0);
#4= move(delta_x= -10.0, delta_y= 0.0);
#5= move(delta_x= 0.0, delta_y= -4.0);
```

Traditionally this type of vector representation is intended to control plotter devices. As a result (2D) shapes are broken down in series of straight lines (with varying line lengths to mimic curved lines). Because of this purpose they do not contain any structuring other than poly-line or polygon (for fill area or hatching operations). Zooming in will ultimately reveal this intrinsic structure of straight lines.

- *spatial enumeration*

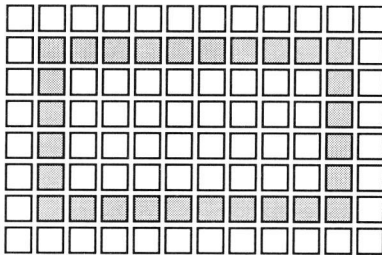


Figure 3.7 Rectangle represented by pixels.

¹¹ For this purpose most algorithms break down each polygon to a set of triangles (triangulation) to guarantee that each polygon represents a flat plane (in 3D space). Additional normal vectors can be used to smooth the faceted surfaces [Neider 1993].

Spatial enumerations (bitmaps) define a rectangular equidistant grid of dots, with associated colour information for each dot. They are intended to be presented on pixel oriented screens and printers (laser or ink-jet). All structure information is completely lost. Zooming in will quickly reveal the intrinsic structure of coloured dots.

Especially the pixel oriented representations are interesting with regard to their semantic content. From a machine perspective the information content is virtually nothing except a grid of coloured dots. However, from a human perspective this same grid could contain a rich semantic content, e.g. a scanned technical drawing. This type of representation is sometimes referred to as *presentation*, i.e. a representation that is intended for human interpretation only. This definition includes besides pixel oriented representations also representations generated by word processors, e-mail applications, web browsers/servers and slide presentation packages. In other words applications that represent the major usage of computers, especially PC's, today.

A technical drawing is also a presentation: a representation that is intended for human interpretation only, even if it uses vector representation instead of pixel representation. The next section will explain that this simple fact is the main reason that digital exchange of technical drawings will not improve computer integrated construction.

The remainder of this chapter will focus on the shape information exchange between humans and computers in the construction industry. The next section will discuss the historically developed dual role of shape information to represent a shape *and* a reference frame to attach product data.

3.2 Shape as the Carrier for Product Definition Data

The realisation of a complex product like a building is only feasible with a reliable information system. Before the age of computers this information system was a complex of data distributed over:

- physical information exchanges and archives tied with paper-based carriers¹² like drawings, reports, order forms, contracts, minutes, memo's, notes,
- auditory exchanges in briefings, meetings, telephone calls,

¹² Even a 'modern' communication device as the fax belongs in this category of paper based information exchange.

- ‘meta’ information in building codes, building regulations, building standards,
- and last but not least the distinct informal ‘information models’ residing primarily in the minds of the participating humans representing various disciplines (architectural model, structural model, energy model, fire safety model, and so on).

The introduction of computers affected the physical information representation thoroughly. Nevertheless, the level of information representation was still aimed at human interpretation, i.e. although now partly produced using computers, drawings and reports remained the basis of the actual information representation.

3.2.1 Drawings

Drawings are still the main information carriers for the representation of the engineering view on product definition data. Drawings stand for much more than just a set of orthographic projections and cross sections of the shape of a product. Because of the additional meanings behind things like line styles, hatching styles, symbols, annotations and dimensions a drawing has more the function of a graphical and symbolic language, comparable with the mathematical notations. Over the years this ‘drawing language’ has been developed and standardised and is, of course, principally meant to support the communication between humans.

The efficiency of the information processing system will not increase dramatically if a computer aided drawing system replaces a drawing board with the objective to produce the same drawings as before (besides a faster and higher quality drawing production)¹³. For a substantial improvement the information must be fit for communication between computers. However, with regard to the state-of-the-art in computer science, drawings (even in digital format) are very hard to understand by machines. For instance, to generate a genuine three-dimensional model out of several orthographic projections is a complex process that is not always robust because of ambiguities or absent data. Humans will not be fooled by these small inconsistencies, yet, machines will¹⁴.

¹³ This is a similar development as the transition process in text production from type writer to word processor. Again the improvement is more in terms of quality rather than speed.

¹⁴ However, real inconsistencies between the various parts of a technical drawing could survive until it becomes manifest in the realisation phase.

3.2.2 Three Dimensional Models

An obvious step to take seems to represent the shape of the building object directly by a three-dimensional model instead of a set of two dimensional projections. Until recently, physical scale models (mock-up) were the only available 3D models. Computer science has contributed the possibilities of digital 3D shape representations (wire modelling, surface modelling and solid modelling).

Shape definition using a 3D model has several advantages over a set of 2D projections and cross-sections:

- *more consistent shape definition*
Inconsistencies between the various 2D projections of the same shape may occur easily because of the redundancies of the applied representation. Sometimes these errors are recognised in a very late stage and, because of that, costly to fix. A 3D model can be represented practically without redundancies.
- *more complete shape definition*
Traditionally 2D drawings skip certain parts of the shape definition of a building object (assembly) which are hard to represent, for instance building knots. 3D models are able to represent these building knots, although this appears also a disadvantage because it means additional work and, unfortunately, it is a difficult job. This is a typical example of how flaws of the information system are traditionally solved by delegation of design details to the contractor¹⁵.
- *superior in complex geometry*
Complex geometry, e.g. curvatures in two directions, is very hard and occasionally impossible to represent adequately in 2D drawing representations. Again 3D models are capable to represent virtually any surface, although it can be difficult to get it right from behind a 2D screen. Sometimes a solution is found by constructing a scaled wooden model or clay model¹⁶ (which is of course again a *physical* 3D model representation). In the aircraft and automotive industry, where double curved surfaces are rather common, physical models and digital 3D models are used concurrently with transitions in both directions. An infra-structural object like a highway is an example in the building industry with a rather complex 3D geometry. Consequently, applications for highway modelling typically use a mixture of 2D

¹⁵ Actually delegation of design is in general not a bad thing, it improves the flexibility in the manufacturing process. The point is that delegation must be a rational choice and not the forced result of imperfections in the product information representation.

¹⁶ This can also be done automatically using techniques like numerically-controlled milling or the stereolithographic laser technique.

and 3D representations. Nevertheless, the in essence one-dimensional sweep-like character of the shape of a highway makes it still fit for the final representation in annotated two-dimensional projections and cross-sections.

- *fit for co-operative design*

Two-dimensional representation of a three-dimensional shape is principally a derived shape representation, i.e. the actual 3D point set is mapped into a 2D domain with a manifest loss of data. The reverse transition, retrieving a 3D point set from 2D data, is mostly not possible with exceptions for shapes with a relative simple geometry (preferably flat and straight). This means that if a 2D representation is applied for the data exchange of a shape design the receiving party is not able to make any essential modifications. Yet, keeping the exchange at a genuine 3D representation level is a fundamental requirement to support co-operative design.

3D modelling has also a few disadvantages:

- *more work*

Defining a consistent 3D model is often more work than, for instance, just to draw a plan view plus annotations.

- *drawing generation*

Drawings play and will play a vital role for the communication with and between humans. A 3D model is actually only presentable on a computer screen. A perspective view is nice, but eventually a drawing must be generated. Fully automated drawing generation is not attainable, that is there is always some editing needed, e.g. for adding annotations and dimensions¹⁷.

Generating a drawing from a 3D geometric model will reveal once more that drawings do contain (much) more information than just a representation of the shape of the building object. The 3D model must be augmented with additional product data with regard to the

- applied materials,
- surface blending,
- finishes or coatings,
- explicit dimensions,
- references to standard parts,

¹⁷ However, this problem is much smaller if the drawing is presented on a computerscreen than on paper. A screen presentation can be configured to show certain aspects, while hiding others (enabled and disabled layers). Besides, the user is able to retrieve information by selecting a specific drawing object with for instance a mouse click.

- realisation recommendations,
- conformance requirements and so on.

Current practice in the construction industry demands drawings for the major exchange of product data between design stage and manufacturing stage (formal specifications). Therefore, two important obstacles hinder a transition from 2D shape representation to 3D shape representation:

- the need to generate in the end drawings from the 3D model, and
- to augment this 3D model with product data.

To augment a 3D model with product data is not as simple as (maybe) expected. 2D drawings are meant to support human-to-human communication, which means that the product is incompletely represented because the absent data can be filled in with common-sense reasoning. 3D models are meant to support machine-to-machine communication, which means that the product must be entirely and unambiguously represented. To accomplish this completeness the sending and receiving applications should agree about the way the information exchange is structured. In other words they have to share a conceptual information model (sometimes referred to as product model) for this type of exchange. Obviously, standardisation of such a conceptual model is inevitable to bring the necessary efficiency to make this new technology successful. Various standardisation efforts have demonstrated the complexity of this approach:

- *agreement on the 3D shape representation scheme*
Numerous 3D representation schemes are available. Which to choose is not trivial and often dependent of the kind of application that needs a shape representation to operate upon. In other words there is no such thing as *the shape representation* of a building object.
- *agreement on the product data to augment the 3D model*
This causes a lot of discussion about what concepts, classifications and so on, to use (often dependent on local or national practices) and how to define the exact semantics of each concept.
- *agreement on how shape data and product data are related to each other*
Should a certain concept refer to a solid, a surface, a line or a point? Or sometimes more detailed: a loop, or a shell, ...? Evidently, this point is directly related with the choice of shape representation.

3.3 Product Modelling

This last point brings up the issue whether a 3D shape representation offers the ideal data structure for organising product definition data. Apparently, such a data structure was not designed for that purpose in the first place. Indeed shape representation schemes often show a flat structure (B-rep) or a sequence of point set operations (CSG), where product data is better structured according to a hierarchical scheme. In addition, it is often awkward to be forced to define a shape in advance *before* any product data can be stored.

These insights introduced the next step in product data representation: product models. Unfortunately, there is hardly consensus about the definition of what a product model is. It may vary between a data structure with the capability to attach non-geometric functional data to a shape model until a rich product structure with the capability to attach various shape models. An important research field as feature modelling is probably positioned somewhere in the middle of this scale. Feature modelling is a kind of equilibrium between geometric data and functional data [Bronsvoort and Jansen 1993].

An advanced (i.e. a shape neutral) product model concept is chosen for this thesis. The reason is to draw a clear distinction between all geometric model structure based approaches for storing product data and a shape neutral approach. A product model is defined here as an information model that is able to store, in an application and geometric neutral format, all the information of a certain product during its complete life cycle¹⁸. Shape representation is no longer a candidate to deliver a structure for such a (necessary complex) model. On the other hand, a single product model may bundle many different shape models. Shape models dealing with various levels of detail and different applications (visualisation, structural mechanics, energy calculations, bill of materials, technical drawings, regulation checking, virtual reality, robot control and so on).

3.4 Conclusions

The information that is exchanged between two parties needs a representation that fits with the chosen information carrier and an agreed semantic level. This representation may take many forms, ranging from very terse (implicit) to very expanded (explicit) and ranging from: more fit for human interpretation (presentation), to more fit for machine interpretation.

¹⁸ For instance: as-required, as-designed, as-planned, as-built, as-used, as-maintained, as-demolished.

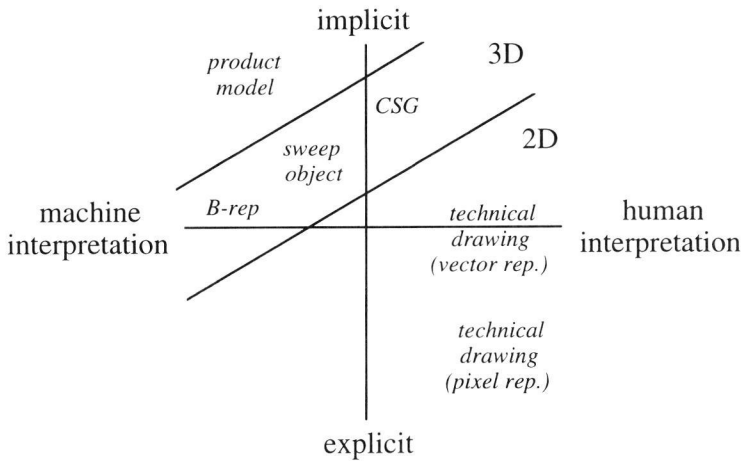


Figure 3.8 Representation forms ordered to their position on the implicit/explicit scale and fitness for human/machine interpretation scale. Technical drawings are typically meant for human interpretation but may be based on a vector representation (normal CAD system) or pixel representation (e.g. the result of scanning of the original paper drawings).

To raise the level of meaning and the quality of the information infra-structure these representation forms should conform to the following requirements:

- *an implicit representation*
An implicit representation offers the convenience to derive more explicit representations, the inverse operation is mostly undefined.
- *a machine-interpretable representation*
Only with a machine-interpretable representation genuine integration of computer-supported processes can be realised.
- *a shape independent representation structure*
Only a shape independent representation structure is able to support multiple shape models for the same object and to offer a backbone for all other, often shape independent, properties.

Candidate representations are:

- *2D drawing*
The 2D drawing representation is rejected because it fails all the requirements, i.e.:

- it is not a real implicit representation
this can be demonstrated by the complications to derive a 3D model from 2D projections
- it is not intended for machine-interpretation
obviously because of historical reasons
- its structure is not shape independent
a technical drawing can be produced only after a certain stage in the design process where many shape decisions can be regarded to be relatively stable.
- *3D model*
The 3D model representation is better equipped with regard to implicitness and machine-interpretability, however, it is certainly not shape independent.
- *product model*
A product model may in principle meet all these requirements.

3.5 Research Questions

In agreement with the conclusion that a product model may in principle meet all the requirements for raising the level of meaning and the quality of the information infra-structure, the remainder of this thesis will be based on product modelling as the preferred representation. This choice partly answers research question 1:

1. *Which product (and process¹⁹) representation is able to meet the requirements of the construction industry?*

However, the choice for product (and process) modelling is primarily founded on the conclusion that other candidate (shape related) representations have manifest disadvantages. For product modelling to fulfil its promise, i.e. an application independent data structure that is able to store and to retrieve all the information of a certain product during its complete life cycle, it will be obvious that especially the structuring part turns out to be crucial.

Therefore, research question 1 is further detailed by questions 1.1 and 1.2:

¹⁹ As will be discussed in the next chapter product modelling and process modelling are strongly interrelated. This raises a problem for the development of 'pure' product models or 'pure' process models. However, if this interrelationship is accepted as a given fact it is more effective to model product data and process data in one and the same reference model.

1.1 *Which structuring mechanisms can be employed to organise a product (and process) model?*

1.2 *How can these structuring mechanisms be integrated in one super-structure?*

The next chapter (Product Modelling Structures) will discuss the various requirements for a widely applicable product model data structure.

4

Product Modelling Structures

How do we find a specific piece of information in a large and probably unfamiliar product model? Or phrased differently how do I store data in a product model that can be retrieved by others and by myself? This question is not sufficiently solved with standard database data retrieval techniques because familiarity with the conceptual model of a specific product model may, in principle, not be presumed. Obviously we need a super-structure that persists independently and invariantly of any specific conceptual model and its instantiations (product models).

4.1 Introduction

In the ‘conclusions’ section of the previous chapter it was expressed that shape related representations, in principle, are not able to fulfil the requirements for an implicit, machine-interpretable and shape independent representation structure. Hence, we need a more general product model structure. However, this statement only asserts the potential capabilities of the product modelling approach. Without any guidance, product modelling could easily lead to numerous parallel initiatives to develop models ‘from scratch’ resulting into a pile of incompatible models and a massive fragmentation of resources. However, to turn it into a successful technology, product models should comply with a set of requirements that guarantees a certain degree of structural homogeneity and therefore genuinely raises the semantic content of product data exchange.

Of course, this insight is not new and was already recognised at the start of the ANSI/PDES²⁰ initiative [Smith 1986] and later adopted by the ISO/STEP²¹ standard [ISO/TC184 1994a]. It is a pity that the final ‘core model’ of STEP (part 41 [ISO/TC184 1994b]) has very little semantic content and offering not much

²⁰ PDES: originally the acronym for Product Data Exchange Specification and meant as a follow-up of IGES (Initial Graphics Exchange Specification). Later the PDES initiative was fused with STEP, changing the acronym in: Product Data Exchange using STEP.

²¹ STEP: STandard for the Exchange of Product model data. Informal name for IS 10303 *Industrial automation systems and integration - Product data representation and exchange*.

more than a network graph with explicit associations. An influential model, which even was incorporated in the first draft version of STEP, is the General AEC Reference Model (GARM) [Gielingh 1988]. In spite of its name this model addresses a more general level than the AEC²² industry. It recognises several important product data structuring mechanisms, such as:

- the distinction between required, proposed and realised properties,
- modular decomposition,
- the specification levels: generic, specific and occurrence,
- reference geometry

The GARM affected a number of other reference model developments such as:

- Road Model Kernel, a highway design data model developed by the Dutch Ministry of Transport, Public Works and Water Management [Willems 1990a]. This model was a first approach for a so-called product type model, a conceptual model with a limited scope (i.e. not 'all products' or 'all mechanical engineering products' but a well defined sub-set or product family, here highways). An interesting feature is the application of the GARM structuring mechanism of distinction between required and proposed properties as a configuration mechanism to switch between various plug-in sub-models.
- IMPPACT, Integrated Product and Process Modelling of Discrete Parts Manufacturing (ESPRIT project 2165). The IMPPACT Reference Model [Gielingh and Suhm 1993] focuses on the relation between product information and activity information for the mechanical industry. It introduces the idea of explicit orthogonal modelling dimensions, which appears to be a powerful structuring technique to help classify the available concepts of a universe of discourse. The model has also successfully demonstrated the use of specialised product type models for sheet metal parts and complex shape parts, like ship propellers.
- NIDDESC, US Navy-Industry Digital Data Exchange Standards Committee, now incorporated in the MariSTEP Program. A set of models dealing with various aspects of shipbuilding (structural, outfitting, raceways). Especially the GARM mechanism of distinction between type and occurrence information is adopted in these models.
- COMBINE, COmputer Models for the Building INdustry in Europe. Research project within the JOULE programme. The Integrated Design Model (IDM) is a conceptual model for buildings that is optimised for energy calculations. Reference geometry is elaborated in particular in this model.

²² Architecture, Engineering and Construction.

- PISA, Platform for Information Sharing by CIME Applications (ESPRIT project 6876). The Project and Process model [PISA 1996] is a high-level generic reference model, i.e. not aimed at a particular type of industry or specific life cycle stage. It can be regarded as a generalisation of the IMPPACT Reference Model.

Other example reference model developments are:

- CIM-OSA, Open System Architecture for CIM (ESPRIT project 688). Introduced the Enterprise Levels: Business, Application and Physical [OSA 1989],
- MARITIME, Modelling and Reuse of Information Over Time (ESPRIT project 6041). Maritime introduced the modular building block (bricks) approach and was an important input model for ISO/STEP Application Protocol 218 Ship Structures (AP 218).
- CIMsteel, Research project within the EUREKA programme [CIMsteel 1995]. The LPM (Logical Product Model) is the CIMsteel equivalent of a product type model. It tries to integrate different actors involved in the creation of steel structures,
- ATLAS, Architecture, Methodology and Tools for Computer-Integrated Large-Scale Engineering (ESPRIT project 7280). The LSE model [Tolman, Böhms, Nederveen and Bakkeren 1994] introduces the view models that describe the various building systems (space, distribution, structural and separation), and details how to interrelate the so-called inter-system relations,
- Building Construction Core Model (BCCM) [Wix, Tolman, Poyet and Björk 1994]. The BCCM aims to offer an integration structure for all AEC application protocols in STEP and can be regarded as elaboration of the ATLAS LSE model for the building and construction industry sector,
- Industry Foundation Classes (IFC), has used the BCCM as a starting point for a new initiative to create an industry standard (outside the ISO/STEP standardisation) for the building and construction industry. A consortium, the International Alliance for Interoperability (IAI), divided over a number of national chapters is responsible for the development,
- GEM, Generic Engineering analysis Model (ESPRIT project 8894). The GEM core model [Leal, Böhms, Cazeaux, Goult, Helpenstein and Korwaser 1997] aims a similar position as the BCCM, i.e. integrating all engineering analysis data needs of STEP application protocols. It will be further developed as an application resource model within STEP (100-series),
- EPISTLE, European Process Industries STEP Technical Liaison Executive, conceptual model of process plants [EPISTLE 1994] developed by a consortium of process industries. The EPISTLE model is based on the concept of

associations or objectified relationships: independently instantiated association objects that offer much modelling freedom but lack a manifest model structure. It forms the basis of various other model developments, e.g. the STEP application protocol for process plant design (AP221) and the POSC (Petrotechnical Open Software Corporation) model for the oil and gas industry.

This list of model development projects is an incomplete overview of the numerous attempts to implement conceptual models aimed at certain regions in the vast modelling domain that can be spanned by axes for

- generic vs. specific,
- aggregation level,
- type of industry,
- type of application,
- type of discipline (views),
- life cycle stage,
- ...

Although the potential modelling domain for product data is vast, the mechanisms that are needed to structure that data, i.e. to give each piece of data a unique and retrievable address, are probably limited. It would be a major step forward for product data technology if such a set of structuring mechanisms (sometimes referred to as STEP Framework [Kirkley and Seitz 1991]) were agreed upon. A framework like this could help to organise the different developments by offering layers and slots for each possible model (irrelevant if it is already implemented or still non-existing).

In this thesis, that focuses on the integration of product modelling and shape modelling in the construction industry, product data structures are especially important to define the semantic context of a particular shape model. For example, the question whether a shape model is meant as global or detailed can be derived from the location in the product model structure (and therefore the context) this particular shape model is referenced. This chapter will investigate a set of structuring mechanisms that are specifically important for this purpose, i.e. defining semantic context for shape models. After a description of shape model structures in chapter 5, the actual relationship between product modelling and shape modelling will be elaborated in chapter 6.

4.2 Objectives for product data structures

Because the extent of the information describing certain complex products (e.g. a large building) could be tremendous, it will be obvious that especially the structuring part of a product model plays a vital role in product data technology.

Objectives for product data structuring are:

- a backbone so that each information item can be reached from a given root entry,
- modularity for partitioning the information to facilitate parallel access (concurrent engineering), references to (standard) library parts and distributed data storage (network architecture),
- facilities for data sharing, inheritance or reuse for ease of maintenance,
- facilities to distinguish between the model itself and its environment.

Product data structure will generally utilise various mechanisms to classify the total product information. Classifications may refer:

- to the type of relations between the objects (e.g., decomposition, connectivity, or property relations), or
- to the information content of the object itself (e.g., as-required, as-proposed, as-used), or
- to the manner the data is stored: definition, representation, presentation.

To prevent a laborious integration effort *after* the development of a particular product model (bottom-up approach), it is better to develop a model in conformance with a generic reference model that incorporates the principles needed for model structuring (top-down approach).

This chapter outlines these required structuring mechanisms for product and process modelling. Process modelling is brought into the interest domain because information exchanges in practice often appear to contain both product and process data in an integrated form. Activity modelling methods, like IDEF0 [SofTech 1981], have already stressed this mutual dependence relationship. Especially in the realisation phase of a construction project it is rather artificial to make a plain distinction between the product in the making and the process(es) that effect this making. At the end of this chapter a reference model for product and process modelling is presented, that incorporates these required structuring mechanisms. It is developed during the research for this dissertation and is adopted by the PISA ESPRIT project.

General reference models or core models have always been a controversial topic in the debate around product modelling issues. One reason for rejecting the use of reference models are the massive constraints these models impose on

the modelling freedom, forcing to instantiate many information objects even for simple information transfers. An example at the end of this chapter will show a technique to relax these inflexible restrictions.

The next sections will discuss a set of product and process modelling mechanisms that must be provided for as prerequisites to be widely acceptable as a general reference model for product and process modelling.

4.3 Product, process or peripheral

A key question in conceptual modelling is always which concepts should be in-scope and which not. Poorly defined models typically reveal an unbalanced mix of very detailed partitions and very shallowly defined or completely missing partitions. In conceptual models dealing with product and process modelling it seems rather trivial that ‘product’ and ‘process’ should be the two top or root concepts of any model. However, a conceptual model mirrors real world entities and the real world does not make a distinction between in-scope and out-of-scope entities. A conceptual model does (or better has to) make this distinction to prevent it to model-the-whole-world. Therefore this first modelling mechanism requires discrimination not only between information that describes a certain state (product) and information that describes the transition from one state to another (process)²³. However, it should also deal with information from environmental agents that influence certain properties of the in-scope concepts yet emerge from concepts that are themselves out-of-scope (peripheral).

4.3.1 The association between products and processes

Product and process modelling are strongly interrelated: a product is always the result of a process, while a process needs input products to transform, to aid the production, or simply to consume²⁴. Especially during the production phase the extent of the information needed to manage the processes is at least as important as the amount of information describing the associated products (input, output and intermediate). Furthermore, process modelling concepts have much in common with product modelling concepts, because processes:

²³ Mind that for each name of an entity one should read ‘information describing a ~’, especially a term like *process* gives rise to dynamic associations.

²⁴ This may seem a trivial statement, however in many projects product and process models are handled in isolation, without much attention for interrelations.

- have properties,
- may be decomposed into sub-processes,
- have life cycle stages (i.e., processes must fulfil requirements, have to be designed, planned, performed and terminated),
- can be connected (e.g., to perform critical path calculations or for planning the allocation of resources),
- claim portions of time and space.

As a rule of thumb real product and process models will typically describe the internal process chain of an enterprise with its production process as root entity. Pure product models are more likely to be communicated between companies in a project partner association or in a commissioner/subcontractor relationship²⁵.

4.3.2 The association between in-scope and out-of-scope concepts

During the development of an information model it has to be decided which type of concepts will be modelled in this context, hence declaring these concepts in-scope and also implicitly declaring all other concepts out-of-scope. In the ideal situation all in-scope concepts have only relevant relationships with each other (closed model). However, the common situation is that essential relationships with out-of-scope concepts cannot be ignored. For this reason besides the key in-scope concepts product and process it would be convenient to be able to refer to an out-of-scope concept in the direct environment of the model: the peripheral concept. The peripheral concept is a place holder to model the influence of the out-of-scope products or processes of the environment on the in-scope products or processes of the model itself. The nature of a peripheral is to compensate for the effects of the disconnection of the model and its environment on the behaviour of the model. For example, if a beam is in-scope but the support structure is out-of-scope, then the influence of this support structure can be compensated by a peripheral that models the reaction force characteristics of the support structure, reducing it e.g. to a spring object. Typically, a connectivity relation can model this influence between the peripheral and the product(s) or process(es) that are directly affected by this influence. The out-of-scope character of a peripheral entity should probably reduce the set of modelling concepts that can be applied. Decomposition into sub-peripherals or connectivity between peripherals seems less appropriate, however, it is difficult to provide universally applicable rules here.

²⁵ However, in the construction industry this distinction is not particularly apparent. A better view is obtained with the endorsement that in a construction project the partners form a temporary company (virtual enterprise).

4.3.3 Shape aspects of products, processes and peripherals

Most products claim a more or less static (depending their internal degrees of freedom) region in Euclidean space to host its material realisation. Sometimes the enclosing shape contour is more appropriate, e.g. for products with another physical aggregation state, like liquids and gasses.

The ‘shape’ of a process is a less obvious phenomenon. Still processes definitely claim a portion of space during various stages of their life cycle. A particular important example is the minimal space that is required to put parts together in an assembly. Other examples are:

- the arrangement of a building construction site
- material storage facilities
- transportation clearances
- temporal structures (scaffolds, supports, moulds, lifts, cranes)
- human resources facilities (sanitary, canteen, parking lot)

The shape of a peripheral is important if it imposes a spatial constraint on the in-scope products or processes. Examples are:

- a neighbouring building
- an existing road
- spatial claims of possibly out-of-scope systems (e.g. HVAC or electrical)

4.4 Specification

Information can be specified at different levels. A level determines the scope or the range of information objects that are affected by that particular specification level. In many cases this influence domain is limited to precisely one information object, but for various modelling reasons it may be necessary to expand this domain by sharing the information between many information objects.

4.4.1 Generic, type and occurrence specification levels

Three fundamental levels can be distinguished although sub-hierarchies at each level may create in-between levels:

- Specifications on the occurrence level typically affect only one information object. An information attribute that has a different value for every object (and the fact that two objects have the same value is regarded as coincidental) should be specified at this level. A classic example are the location and orientation attributes for objects in a geometric model.
- Specifications on the type level affect all the occurrence objects that depend directly or indirectly on a particular type object. An information attribute that has the same value for groups of objects and that fact is considered intentional, should be specified at the type level. A corresponding classic example is the shape specification that all occurrences of the same type must share.

Type objects may build up a parent/child hierarchy. The semantics are that a child object inherits the attribute values from its parents (value inheritance), although parental type objects not necessarily specify a complete value set. The missing values can be specified explicitly for the child type object, while an inherited value may be replaced by another value.

- Finally a generic specification level can be distinguished. This is a meta-level, i.e.: data describing data or in other words the generic specification level is identical to the information modelling level defining entities and attributes of entities. In figure 4.1 the entity defining a rectangle by two dimensioning attributes is a generic specification. Generic specification entities may also build up a parent/child hierarchy, although inheritance rela-

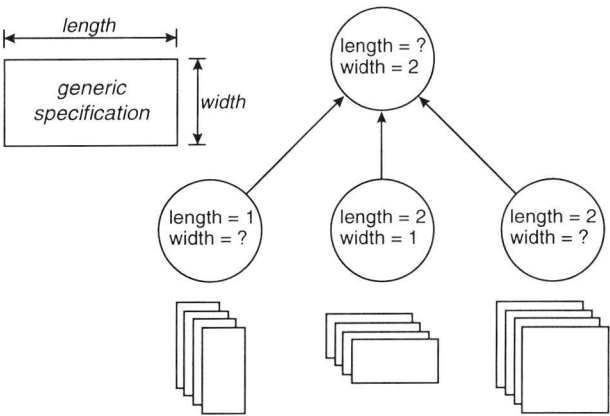


Figure 4.1 Hierarchy of type objects showing the resulting value sets that can be referred to by occurrence objects. A question mark denotes no explicit value assignment; the arrows represent the value inheritance relations, pointing to the parent.

tions now pass on attribute *definitions* instead of attribute *values*.²⁶

The generic specification level needs not to be defined explicitly in a product and process reference model, because this meta-level is an inherent construct in any information modelling language or object oriented programming language.

The distribution of specifications over several levels that differ in scope is sometimes considered as simply a technique to reduce the size of models that show much repetition in their information objects. However, there is also a semantic perspective to it: specification levels stress the fact that objects having the same values is intentional and not coincidental. The specification levels may therefore mirror the decision levels of the organisation that created and updated this particular product model.

An occurrence object does not always capture the information of a 'real world' object in a one-to-one relationship (absolute occurrence). In many models the occurrence objects are real occurrences with regard to a kind of assembly object but still represent groups of real world objects from the perspective of the model as a whole. The section dealing with decomposition will discuss this phenomenon.

4.4.2 Shape aspects of specification levels

Specification levels are easily mirrored to shape models. More over, examples to demonstrate the application of specification levels often use shape related cases. Shape applications may apply user defined parametric objects (generic specification), object creation defaults or copy object attributes facilities (type specification), while the normal creation mode reflects the occurrence specification level.

However, specification levels for product modelling and specification levels for shape modelling will normally differ in their objectives. Shape model specification levels will only optimise for ease of creating this particular shape model, while product/process model specification levels try to reflect the decision structure for a certain design, configuration, production process, and so on.

²⁶ Although the use of constructors may also affect the initial value set of a newly created entity instance.

4.5 Decomposition

The prime objective of decomposition is to organise the information objects with respect to the level of detail they represent. The classical approach is to distribute the information objects in a tree structured graph, where the root node represents the total scope of the model and the leaf nodes represent the finest granularity of detail. Each intermediate node, i.e. a node that has both child nodes and a parent node, represents the assembly of all its siblings. At the same time, it is a sibling of a node that represents a more global level of detail.

An assembly/part structure is another way to contemplate decomposition. In an assembly/part structure atomic parts are the building stones to constitute sub-assemblies, which in their turn are parts in a more complex sub-assembly until a final top-assembly is reached. This metaphor also demonstrates that sub-assemblies never overlap. Each part has only one parent and can be uniquely addressed by specifying the path of sub-assemblies to arrive at that part.

There is no such thing as *the* decomposition of a product or a process. Given the atomic information objects several decomposition trees are not only feasible but may also be very relevant. Especially the life-cycle stage is an important bias for selecting a particular decomposition structure. For example, a designer will usually apply another break down than a manufacturer. The classical illustration is the space structure approach of an architect versus the material structure (space boundaries) approach of an engineer/contractor. It is complicated to realise (and maintain!) multiple decomposition structures in one model. A feasible implementation is to store only one decomposition structure and emulate the other structures, or at least not allowing more than one decomposition per life cycle stage.

4.5.1 Decomposition structures

A fundamental issue is the relationship between the objects in the ‘real world’ and the data that must be kept with respect to those objects in an information model. In principle there should be a one-to-one relationship, but in many situations this is not necessary or even desirable. For example, in the design stage of a car the information concerning the design of a wheel is stored only once, although this part will be applied at least four times. If it is necessary to point out that those wheels are intentionally identical (as it will generally be the case), then it is desirable to store the wheel only once instead of four times. Modifications on the wheel design will then automatically affect all instances of the wheel.

Under other circumstances it could be inevitable to keep a record of each indi-

vidual real world object. For a car in its operational stage it will be necessary for maintenance reasons to store data about each wheel occurrence separately (type of tyre, profile depth, balancing characteristics, ...).

If a one-to-one relation between model and real world is desired decomposition will take the appearance of a pure tree-graph exclusively filled with occurrence objects. Because of this one-to-one correspondence these occurrence objects are referred to as absolute occurrences and the decomposition structure as occurrence decomposition. If there is no need for a one-to-one relationship several decomposition structures are applicable, which can be classified to either:

- type decomposition, or
- mixed type/occurrence decomposition

A type decomposition structure can be derived from an occurrence decomposition by substitution a type object for each group of identical occurrence objects. As a result the tree-graph of occurrence nodes collapses into a directed a-cyclic graph (DAG) of type nodes²⁷. A type decomposition is a very compact

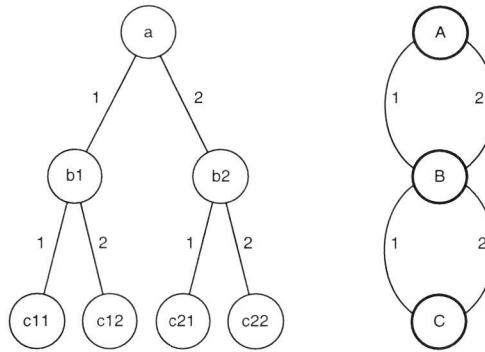


Figure 4.2 Occurrence decomposition (left) with an equivalent type decomposition (right). The type DAG has four different paths from A to C (1-1, 1-2, 2-1, 2-2), which correspond with the four c-objects in the occurrence tree.

structure that exploits the decomposition relations between the type objects as occurrence representations. Occurrences in the real world can be addressed by stating the path from the root to its type object: as many different paths can be stated as many absolute occurrences there are. Because of these properties a type decomposition is the structure of choice for administrative applications like the generation of a part list or a bill of materials.

²⁷ Transforming an occurrence decomposition into a type decomposition (collapsing) has also an inverse operation: expanding a type DAG into an occurrence tree. Because these operations are in fact elementary graph operations they are always applicable as long as the graphs are well-formed trees or DAG's.

The implicit occurrence representation in a type decomposition is usually inadequate for more complex product and process models. The requirement to store at least some information at the occurrence level (mostly location information in space and time) has led to several forms of mixed type/occurrence decomposition structures (figure 4.3).

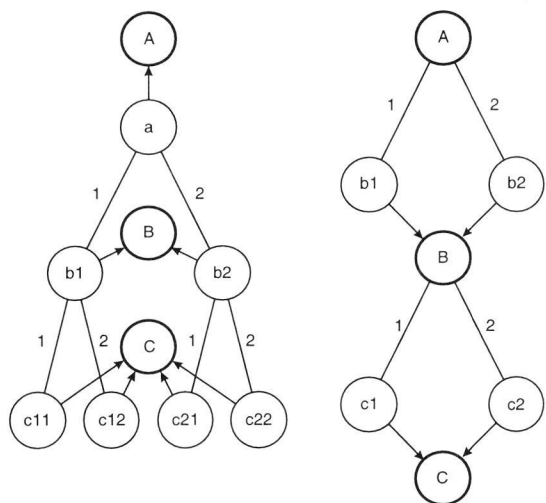


Figure 4.3. 'Enhanced' occurrence decomposition (left) and alternating type-occurrence decomposition (right). The arrows denote the 'occurrence-of-type' relation.

A practical approach is to suppress the redundancy of the occurrence decomposition by bringing in type objects for data sharing purposes. This enhancement does not alter the decomposition structure itself and the one-to-one correspondence with the real world.

Another more radical approach is to objectify the decomposition relations of the type decomposition to occurrence objects. This can be realised by constraining the allowed entity types of the decomposes-into relationship to type and occurrence. Hence, only type objects can be decomposed into a set of, exclusively, occurrence objects. Each occurrence object references a type object, which may be decomposed into yet another set of occurrence objects, and so on.

The occurrence objects in this decomposition structure are relative occurrences, i.e. they have no one-to-one correspondence with the real world. However, they are real occurrences within the scope of the type object in which decomposition they participate. If the occurrence objects contain location data then these co-ordinates refer to the co-ordinate system of their assembly type object. This co-ordinate system is local and has various locations (through the

occurrence objects that reference this type object) in the co-ordinate system of a type object that resides on a more global level of detail. This decomposition structure with alternating type and occurrence objects is a powerful mix of the occurrence objects tree-graph and the type objects DAG. In chapter five this strategy will be used to organise a shape objects graph.

References with regard to this subject, often referred to as *assembly tree*, are [Braid 1978], [Eastman 1981], [Lee and Gossard 1985], [Rappoport 1993] and [Callahan and Heisserman 1996].

4.5.2 Modularity

Modularity is an objective that an appropriate product and process model data structure must accommodate. Many applications can only be accomplished by a modular structure: concurrent engineering, network architectures, plug-in facilities, standard part libraries, etceteras. Nevertheless, a decomposition structure is usually modelled as a monolithic data structure without explicit interfaces. A modular data structure has sufficient one-to-one relationships to disconnect the structure from the rest by cutting the actual relations. The consequence of such a take-out operation should be a set of disconnected data structures that are still valid with respect to the conceptual model.

The selection of the intersection spots or interfaces should keep the internal cohesion of the module at a maximum, or in other words, keep the number of necessary cuts to a minimum. This means for the decomposition structure that it is strongly recommended to keep the assembly object and its decomposed part objects together. According to these considerations almost all decomposition structures discussed in the previous section are monolithic. Only the last one

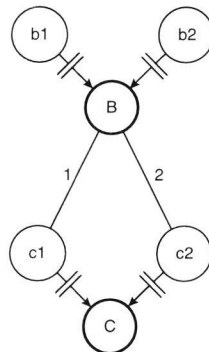


Figure 4.4 Interface locations disconnecting the type 'B' decomposes into two 'c' occurrences data structure to become an independent module.

supports modularity through the is-occurrence-of relations (figure 4.4). This decomposition structure fulfils the modularity prerequisites and it keeps the core objects (assembly and parts) together over one decomposition step.

4.5.3 Concretisation

In several reference models (e.g. GARM) special semantics are associated to modular decomposition structures and considered as a separate modelling mechanism. A natural interface seems to be the link between the specification of the required product or product part characteristics (as-required) and the specification of the proposed/expected product or product part characteristics (as-designed). This interface could represent the link between two different departments of the same company or a contractor/sub-contractor relationship or simply the purchase of an existing product on the market. In a decomposition graph this matching of required and proposed characteristics leads to a structure of further refinement of requirements and further specifying in detail of technical solutions and hence to further *concretisation*.

The bi-typed nodes of the decomposition graph are represented by two prefix qualifiers: *required* and *proposed*. The proposed object plays the assembly role while the required object plays the part role. In other words a proposed object may be decomposed into a set of required objects and a required object can be fulfilled by a proposed object. This last relationship can be utilised as the interface between (dis)connected modules or layers.

Combining concretisation and decomposition in one dimension has always given rise to much discussion²⁸. Yet, there are good reasons for doing it this way. One of the main objectives for product and process modelling is to organise efficiently the available information objects according to the level of detail (or scope) that they represent. This ordering process will lead inevitably to a layered data structure with interfaces between the layers (for modularity reasons), where subsequent layers deal with an increasing (or decreasing) level of detail. The required/proposed interface fits very well in this layered structure: it reflects the responsibilities between departments working at different levels or between a commissioner and a sub-contractor. The required/proposed interface represents the contract between two parties. One party states *what* must be done (and which requirements must be fulfilled) while the other party states *how* it will be achieved (and which characteristics are to be expected). The required/proposed cycle is an essential product and process modelling principle, yet the data structure to realise it happens to be identical with a (modular)

²⁸ This has probably to do with the observation that representatives of certain disciplines either seems to think in terms of functions (e.g. architects) or in terms of solutions (e.g. structural engineers).

decomposition structure. Permitting more than one decomposition mechanism is unacceptable, hence, both modelling principles should be integrated in a product and process reference model.

The required/proposed decomposition structure is popularised by the so called Hamburger diagrams²⁹ that were introduced in the General AEC Reference Model (GARM) [Gielingh 1988]. Again, in combination with the specification dimension (type/occurrence) several data structures are conceivable

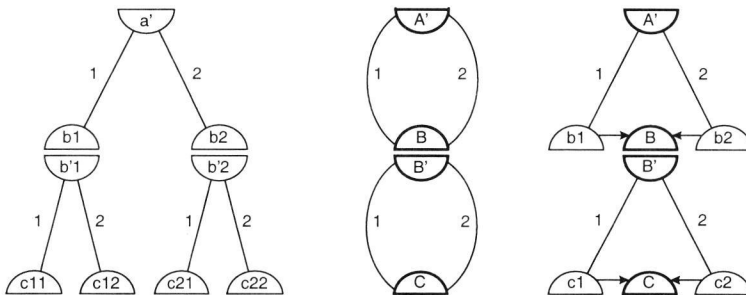


Figure 4.5 Modular decomposition structures: occurrence decomposition (left), type decomposition (middle) and mixed type/occurrence decomposition (right).

(figure 4.5).

With the concretisation modelling mechanism it is now feasible to realise also a modular occurrence decomposition or a modular type decomposition. The mixed type/occurrence decomposition offers two variant required/proposed interfaces: the required *type* is-fulfilled-by proposed type interface (shown in figure 4.5) or the required *occurrence* is-fulfilled-by proposed type interface. In the first case all required occurrence objects, that refer to the same required type object, are fulfilled by only one proposed type object. In the second case each required occurrence object could be fulfilled by a different proposed type object. By introducing a precedence rule both interfaces could be permitted in the same model, e.g., by asserting that an occurrence/type interface overrules a type/type interface. Such a model could mirror systems with global measures and local deviations.

²⁹ Because of the matching semi-circles: the required concept rounded on top and the proposed concept rounded on the bottom.

4.5.4 Shape aspects of decomposition

Decomposition in shape modelling is often simply implemented as a grouping mechanism. A set of shape objects can be collected in a container object, which behaves as a single object for operations as translation, rotation or scaling. A container object may contain shape primitive objects (line, rectangle, sphere, ...) but also other container objects, effectively implementing a decomposition tree. Most shape modelling applications do not allow direct access of an element object after it has become a member of a group. This policy guarantees that each element object of a group cannot be a member of another group, in addition it prevents the possibility of cycles: i.e. a group object cannot contain itself directly or indirectly.

The grouping mechanism will usually be implemented as a non-modular occurrence decomposition. If some form of type decomposition is applied then it will be limited to the user interface: an evaluation algorithm will generate the final occurrence decomposition.

Modularity is seldom implemented, instead most applications offer some kind of 'include' mechanism, i.e. they simply copy the objects of another model in the current model. However, there is at least one exception: VRML (Virtual Reality Modeling Language) [VRML 1996]. This Internet graphics format has an explicit (hyper)linking mechanism to build up a modular shape model³⁰.

Another form of decomposition is exploited in Constructive Solid Geometry (CSG) [Mäntylä 1988]. Here the intermediate nodes not only collect a set³¹ of primitive nodes and/or container nodes, but also apply a set operation (union, intersection or difference) over the operands.

Finally, some cell-based (voxels) shape representations as for instance octree use an intrinsic form of decomposition to organise their cell structure.

Concretisation can be implemented by constraint-based modelling and/or tolerance modelling. This will be elaborated in chapter 6.

4.6 Connectivity

Decomposition organises the information objects over several layers or levels of detail, hence a vertical structuring mechanism. Connectivity determines how the information objects of the same layer are connected, hence a horizontal structuring mechanism. Connectivity is the main structuring mech-

³⁰ The hyper-link approach facilitates distributive concurrent modelling: i.e. partners 'all over the world' could develop a single shape model (distributed over several modules). Furthermore, it will be a so-called 'evergreen' model: there are no copies, each partner renders, using their VRML-viewer, the same model.

³¹ For unambiguous semantics the set size is usually limited to precisely two elements.

anism of product and process modelling. If the decomposition and specification dimensions in a model are ignored (by collapsing all assembly-like objects until only atomic component parts persist and by expanding all occurrence objects with a copy of the entire inherited value set) the relations that remain will be the connectivity relations. The connectivity dimension constitutes a connected network graph of information objects. A connectivity network is a unique structure, i.e. there is no such thing as *the* decomposition structure of a product and process model but there is only one connectivity structure.

Two key node types can be distinguished in a connectivity network: the concept definitions and the property definitions. A concept definition can be regarded as a container object that defines a certain concept through the property definitions it contains (either by value or by reference) and its connections with other concept definitions. A property definition holds the value of some physical phenomenon, which may range between a simple scalar value (e.g. length, mass or temperature) and a complex network (e.g. a shape model³²).

- The concept definitions and their mutual connectivity relations construct the central backbone of a product and process model. The decomposition and specification dimensions do not really alter this condition. They merely transform a single level network into a multi-level graph with tree-like structures between the levels and network structures within each level. In a connectivity network each concept definition is connected, directly or indirectly, to all other concept definitions³³.
- The property definitions are plugged into this network of concept definitions as terminal nodes, i.e. property definition nodes can only be reached through the connectivity network of the concept definitions.

4.6.1 Connectivity network specification

Should connectivity be realised at type or at occurrence level? The answer is that both options are valid (even in the same model), but that the semantics are different. Consider the example illustrated in figure 4.6.

³² Which shows that these terms are context dependent: a shape model will again consist of concept definitions and property definitions.

³³ This property is based on the idea that each model should have precisely one root object, representing in fact the model as a whole. Because decomposition structures are pure trees (in the expanded form of an occurrence decomposition) more than one root object automatically means as many disconnected graphs. Therefore, it seems not sensible to introduce two or more completely independent structures in one and the same model.

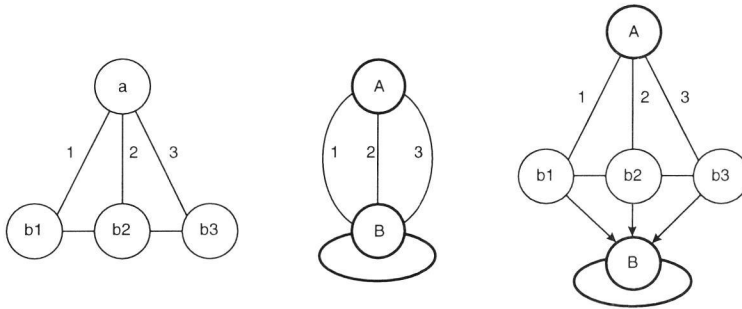


Figure 4.6. Connectivity networks between decomposition components, applying occurrence decomposition (left), type decomposition (middle) and mixed type/occurrence decomposition (right).

The same configuration, i.e. type or occurrence concept object *a* is decomposed into three occurrence concept objects of type *B* interconnected in a chain structured connectivity network, is modelled using different decomposition mechanisms, subsequently: occurrence decomposition, type decomposition and mixed type/occurrence decomposition. Occurrence decomposition models the configuration correct (apart from the inherent shortcoming that the occurrences are intentionally identical). Type decomposition is obviously inadequate to model the chain structure of the connectivity network. Connectivity between type concept objects has no significance unless this relation is interpreted as a *possible or allowed connection type* that constrains the connections between occurrence concept objects. The mixed type/occurrence decomposition applies this interpretation in combination with the actual occurrence connectivity network.

Connectivity between a type concept object and an occurrence concept object (of a different type) is also feasible when it is necessary to state explicitly a cardinality constraint. For example, all occurrences of type *A* are always connected to exactly three occurrences of type *B*. However, this modelling mechanism introduces a kind of implicit decomposition in the connectivity network. Apart from the substantial increase of complexity which is probably not manageable anymore beyond a certain point, it is cleaner to handle decomposition and connectivity as two orthogonal structuring mechanisms.

4.6.2 Modularity

The reasons why modularity is an essential characteristic for product and process modelling are clarified in a previous section discussing decomposition

(4.4.2.). How is a connectivity network practically broken down into separate modules, what are the appropriate locations to make the intersections?

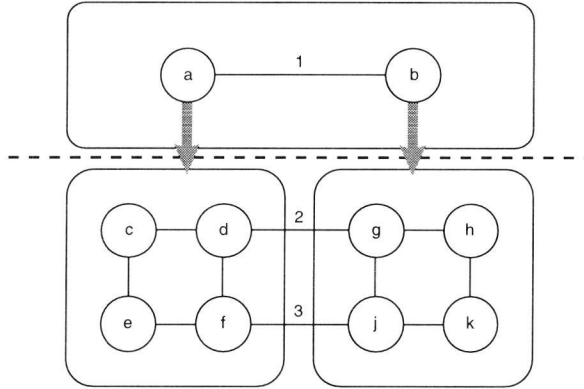


Figure 4.7 Decomposing objects and object relations. Each relation will be a high level interface if the objects at both ends are decomposed in separate modules.

In a flat single layer network the approach to minimise the number of cut relations (i.e. to obtain modules with much cohesion and only a few ‘dangling ends’) could work satisfactorily. However, multi-layered decomposed models complicate this situation, which can be illustrated with a small example (figure 4.7). The figure shows two adjacent decomposition layers: on the higher level two concept objects *a* and *b* and connected by *1*, on the lower level a set of concepts $\{c, d, e, f\}$ forming the decomposition of *a* and another set $\{g, h, j, k\}$ forming the decomposition of *b*. The higher level connection *1* decomposes into two connections $\{2, 3\}$ on the lower level. Obviously, the connections 2 and 3 are excellent candidates for locating the intersections and declaring both sets of concepts to be modules. However, these modules contain the decomposition components of the higher level concept objects *a* and *b*. Genuine modularity demands to make an intersection through *all* the layers³⁴, thus the intersections in the connections 2 and 3 will propagate upwards to affect also connection *1*, implicitly declaring *a* and *b* to be singular object modules. Clearly, the same reasoning leads to intersections in *all* connections on the lower level if a next, yet more detailed, decomposition layer is appended. The conclusion is that each connection in the connectivity dimension is ‘intersectable’ and has to be, because it may be the point where two decomposition branches come together.

³⁴ Until a common assembly object is encountered, i.e. the intersections will assign all branches starting from a particular node in different modules.

A technique to deal with modular (decomposed) connectivity is the incorporation of *ports*. A port object represents a piece of the connectivity interface of the concept object it belongs to. A proper connection between two concept objects utilises two port objects, one for each concept. The actual connection is established by an *interface* object that refers to both ports, one as source and the other as target. These attribute names introduce a directional sense in the actual connection, which may or may not be relevant for certain applications³⁵.

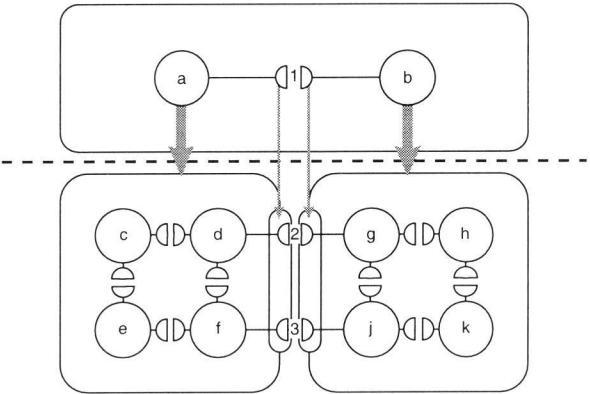


Figure 4.8 Decomposing concept objects, port objects and interfaces. Each interface consists of two port objects that may be decomposed into a set of external port objects.

Figure 4.8 shows the previous example with port objects added. Now every connectivity relation is established by joining two port objects, one from each concept object. An advantageous characteristic of ports is that they stay in place, independent from the fact if the actual connection is established or not. This is an important requirement for modularity: a model with ports is complete, even if it is isolated as a separate module in a part library or in an off-line repository. Any knowledge with respect to the other side of the connection is banned out. If the connection is installed this fact is recorded with the model in which the modules are instantiated and not in the modules themselves.

The example also shows that a concept object can be decomposed into concept sub-objects *and* a connectivity network between these sub-objects. Besides, a connectivity relation can also be decomposed into sub-relations. To control this phenomenon port objects are ‘decomposable’, which is not a special feature but follows logically from the combination of the concretisation modelling dimension (decomposition) and the connectivity modelling dimension.

³⁵ Especially connections between processes will typically show a directional sense by relations like ‘precedes/succeeds’ that determine the temporal sequence. If directions are irrelevant, this piece of information could simply be ignored.

However, a port object only decomposes into *external* port sub-objects, i.e. ports that maintain connections crossing the module boundary, as opposed to *internal* ports: for maintaining connections between concept objects in the same module. The reason for this is that a port is an object to connect a concept object with its environment. The decomposed sub-ports should fulfil a similar function but now for the decomposed sub-concepts as a group.

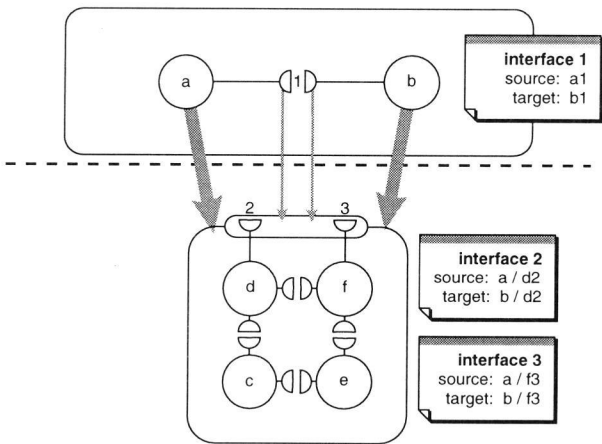


Figure 4.9 Decomposing concept objects into the same module (mixed type/occurrence decomposition). Explicit interface objects must guarantee the correct reconstruction of the connectivity network.

Figure 4.9 shows a variant solution where the concept objects *a* and *b* share the same decomposition, a situation that may occur with mixed type/occurrence decomposition. The concept occurrence objects *c*, *d*, *e* and *f* are now relative occurrences, yet the connections that cross the module boundaries are absolute. They constitute the minimal framework that is necessary to reconstruct the integral connectivity network from a set of shared module connectivity networks. Absolute port addressing is inadequate in this situation. Interface objects must

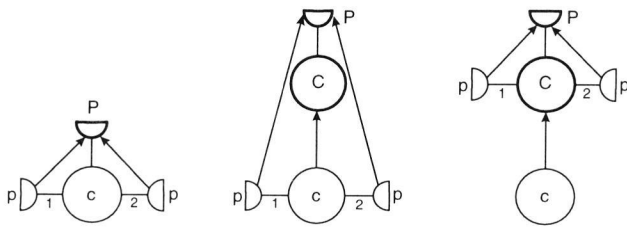


Figure 4.10 Specification level variants for port objects.

employ relative addressing (by stating a path) to specify the absolute connections.

Of course, port objects can be distinguished along the specification dimension as port types and port occurrences. Figure 4.10 shows a number of modelling configurations. Left a concept occurrence declares a port type with a very local significance. In the middle the port type has migrated to concept type level and may be shared by various concept occurrences. At the right even the port occurrences have migrated to the type level. In the last case the port occurrences have changed status from absolute (within the scope of the module) to relative. That means that the interface objects that establish the connections must state port occurrence object *plus* concept occurrence object.

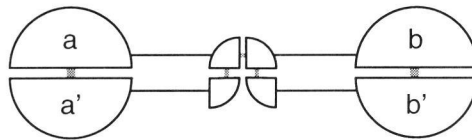


Figure 4.11 Two connected required concept objects *a* and *b* fulfilled by the proposed concept objects *a'* and *b'* respectively. Besides, the required port objects must also be fulfilled by the corresponding proposed port objects.

Modularity with respect to decomposition is easy to integrate with connectivity modularity. As figure 4.11 illustrates the is-fulfilled-by relationship (which is the interface relationship for modular decomposition) is also applied to matching required/proposed port objects. The actual connection is established at the 'required' side of the interface. This choice can be justified by the fact that this side has a more permanent status and because there is no benefit in doubling the connection to both sides.

4.6.3 Properties

The concept definitions of a model constitute the so-called universe of discourse: the set of things that are in-scope with regard to a particular information exchange. However, the concepts and their connectivity relations compose essentially a structure, and this structure must be augmented with property definitions to create a semantically complete product and process model.

Property definitions can be configured using the same structuring mechanisms as concept definitions. Therefore, distinctions can be made between product properties and process properties, between property types and property occurrences, between required properties and proposed properties, etceteras.

However, it is not sensible to attach for example a required process property to a proposed product concept. A better approach is to make the property object dependent from the type of the referencing concept object, e.g. required product concepts have required product properties. This technique also integrates the simple properties, that can easily be represented by an attribute value, and complex properties, e.g. a shape representation, that may be considered as a model in a model. Simple and complex properties are both property objects, although only the last form is actually implemented as a separate entity³⁶.

Spatial and temporal properties are very common in product and process models. Therefore, these properties are often defined explicitly in a reference model. They should be considered as an appropriate place to plug in this type of properties.

4.6.4 Shape aspects of connectivity

Connectivity in shape modelling can be distinguished in internal and external connectivity. Several shape representations need internal connectivity to build up the internal structure or topology of a shape object, e.g., the alternating connectivity between vertices and edges to form a path or loop.

External connectivity, i.e. connectivity between completely defined shape objects is rare in shape models. The reason is probably the still limited application of CAD-systems other than for visualisation or drawing. The absence of external connectivity data cannot be detected by a simple inspection of the shape on the screen or on the plotted technical drawing. However, for numerous other applications external connectivity data is essential: structural analysis calculations, energy transmission calculations, numerically controlled milling/drilling, production planning, maintenance, and so on.

There are two approaches for bringing in connectivity data:

- *reference geometry*
Reference geometry uses a skeleton structure of idealised (reduced dimension) shape objects, e.g. centre-lines and mid-planes act as place holders for the actual (solid) shape. This type of modelling is often found in piping applications:
 - a centre-line may act here as a sweeping guide line for a specific pipe that can be selected afterwards,

³⁶ It is a pity that most information modelling languages make such a vital distinction between entities and attributes, yet some object oriented programming languages (which have often a substantial amount of information modelling features) ease this distinction. An attribute value is either a reference to another object (and maybe shared with others) or an expanded object, i.e. the object is included and cannot be referenced by others.

- a cross-point may stand for a typical pipe connection with as many 'ports' as connecting centre-lines.

Reference geometry can also be applied to support the concretisation modelling mechanism (linking required and proposed concepts). As such it was implemented in the experimental product modeller ProMod [Kuiper 1989], that was based on GARM.

- *non-manifold topology*

To facilitate a flexible connectivity in a shape model an extension of the common topological relations is necessary. For example, besides the bounding relationship between an edge and a vertex, an additional relationship is necessary to allow a vertex to lie 'somewhere between the end-points' on an edge. Non-manifold topology permits these extra relationships.

Compared with reference geometry, non-manifold topology offers a kind of integrated connectivity. On the other hand non-manifold topology is probably the right choice to implement a flexible reference geometry modeller.

4.7 Life cycle

The life cycle modelling mechanism offers in essence a classification mechanism between the three main life cycle phases of a product or a process:

- *construction*
the life cycle phase during which a product or process is under construction, i.e. the structuring process of integrating sub-products or sub-processes (with life cycles of their own) into a more complex product or process.
- *operation*
the life cycle phase during which a product or process is fully operational.
- *destruction*
the life cycle phase during which a product or process is under destruction, i.e. the destructuring process³⁷ of disintegrating a complex product or process into the constituting sub-products or sub-processes.

It is important to realise that the life cycle of a product will usually differ from the life cycles of the constituting sub-products. This means that although the product as a whole is operational parts of it may be destructed during this

³⁷ Using the term destruction with respect to a process is probably not obvious, yet a complex process consists of sub-processes that were possibly already operational before the actual start of the main process and which may continue after its end. Compare for instance all the sub-processes that are needed to bootstrap and shutdown an operating system.

period and other parts constructed. In other words maintenance that is fulfilled by replacing parts with a shorter operational lifetime.

Combining the life cycle and concretisation dimensions offers many slots to distinguish all sorts of life cycle phases:

<i>Product</i>	<i>construction</i>	<i>operation</i>	<i>destruction</i>
<i>required</i>	product under construction requirements <i>needs for manufacturing</i>	product in operation requirements <i>demanded performance</i>	product under destruction requirements <i>recycling requirements</i>
<i>proposed</i>	product under construction proposal <i>design for construction</i>	product in operation proposal <i>functional design</i>	product under destruction proposal <i>design for destruction</i>
<i>Process</i>	<i>construction</i>	<i>operation</i>	<i>destruction</i>
<i>required</i>	process under construction requirements <i>resource allocation</i>	process in operation requirements <i>resource management</i>	process under destruction requirements <i>resource deallocation</i>
<i>proposed</i>	process under construction proposal <i>project planning</i>	process in operation proposal <i>project management</i>	process under destruction proposal <i>deliverables</i>

Table 4.1. Slots for life cycle phases that result from all possible combinations of the classifications of the life cycle dimension and the concretisation dimension.

4.8 A Product and Process reference model

Figure 4.12 presents an example of a product and process reference model. It is developed in co-operation with the PISA ESPRIT project [PISA 1996] and it is able to support all modelling mechanisms that were discussed in this chapter.

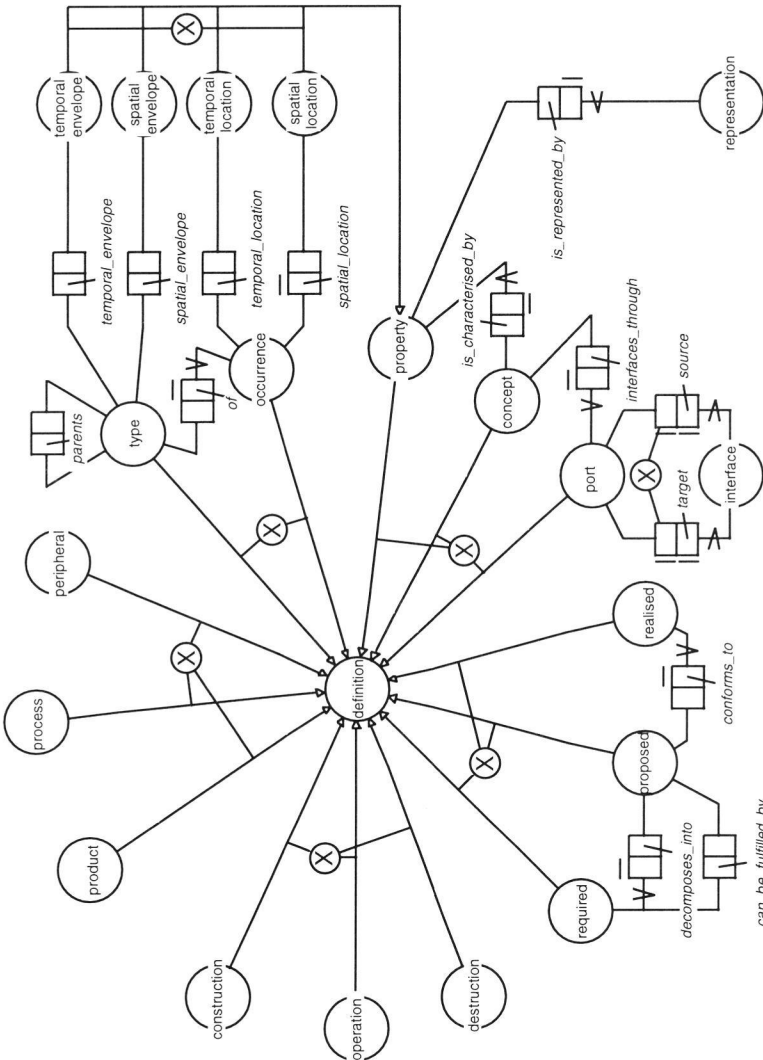


Figure 4.12 NIAM diagram of a product and process reference model.

The product and process model (P&P model) uses five structuring mechanisms that are referred to as dimensions. The dimensions are considered to be

orthogonal, hence, entities can be formed by combining up to five independent entities from each of the five dimensions.

Although a product and process reference model can be applied for direct instantiating, the semantic content is evidently too general to be of much significance. The justification for the existence of a reference model is to ease the integration and the interoperability of product and process models that are developed for a particular branch of industry. The idea is to develop a layered hierarchy of models ranging from broad scoped high level models to narrow scoped low level models all conforming to a common product and process model. In such a framework each slot should contain a model of a well defined and limited scope that could act as a parent model for more specific models or as a server model for other client models. The conformance of the applied structuring mechanisms in each model helps to achieve a kind of plug-in architecture encouraging the reusability of the well-developed models and the replacement of less achieving models.

4.8.1 Model development in conformance with a reference model

How to develop a model that conforms to a product and process reference model? It seems that the answer to this question is not only technological but also psychological or even political. Developers of reference models tend to a traditional but very rigid opinion, i.e. (application) model developers should define all their entities as specialisations (subtypes) of the entities of the reference model. The set of constructs should be used as a kind of high level modelling language on top of the (lower level) information modelling language. Application model developers on the other hand demand flexibility and a freedom of choice between various modelling constructs. The P&P model offers this flexibility in principle, however many information modelling languages are not always able to support this. The most important measure to achieve this freedom is by allowing *not* to discriminate along a certain dimension yet without having to throw away the modelling mechanisms of that dimension. This is not important for the discrimination between product, process and peripheral or between construction, operation and destruction, because these dimensions do not define modelling constructs, i.e. they do not contain defined relationships between the entities of the same dimension. It certainly makes a difference for the other three dimensions:

- The connectivity dimension seems to demand to create a separate entity for each property that is needed to define a particular concept. This could turn out to be a very awkward construct especially if the property can be represented by a single value. However, if the property definition needs a com-

plex multi-valued representation the separate entity construct is very appropriate. Therefore the reference model must allow both constructs, i.e. an ‘internal’ property relation for simple valued representations and an ‘external’ relation for the rest.

- If the requirements for an application model do not demand modularity in the connectivity structure, it is rather ridiculous to model still two port objects for each connection between two concept objects. In this case the separate port objects could collapse to attributes of the connected concept objects³⁸. If interface objects are applied even the separate attributes are superfluous. If the interface objects are ignored the connection can be established by direct linking.
- A mixed approach is also feasible, i.e. direct linking at the lowest decomposition level and indirect (explicit ports) linking at the higher levels.
- If the requirements for an application model do not demand for modularity in the decomposition structure the concretisation dimension entities *required* and *proposed* should collapse to the central definition level making the *is-fulfilled-by* relationship implicit but keeping the *decomposes-into* relationship explicit (figure 4.13).

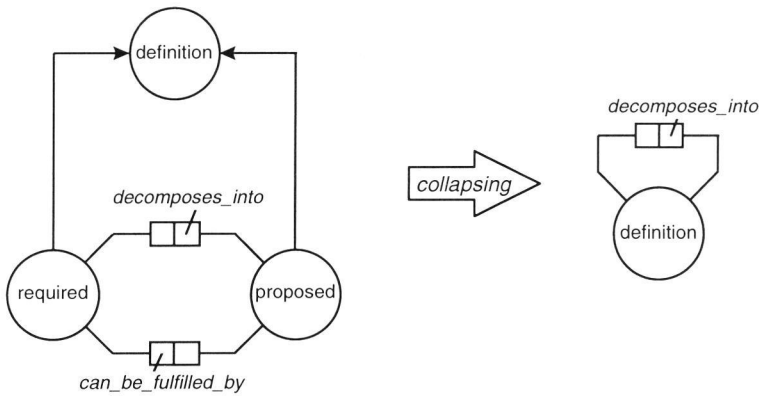


Figure 4.13. Collapsing the modular decomposition mechanism into a monolithic decomposition mechanism.

This ‘collapsing dimension’ approach is almost the same as defining the same relations on the root (definition) entity in the first place. However, by doing that the carefully chosen semantics for each dimension is lost. For example, in this case required definitions could directly decompose into required definitions, proposed definitions or even a combination of both types on the next decomposition level.

³⁸ In EXPRESS one of these attributes can be defined as *inverse* to guarantee the one-to-one correspondence.

- Finally if the explicit discrimination between type and occurrence objects is irrelevant all objects can implicitly be considered as type objects, retaining the advantages of value inheritance. Of course, the different spatial and temporal semantics should merge into the central definition entity.

As already observed, not all information modelling languages will permit the specification of this necessary modelling flexibility. In that case a formal inheritance from the reference model will not be feasible. A second best option can be achieved by labelling the conformance of each entity and each attribute/relation to the appropriate construct in the reference model. This labelling could be done by placing an easily distinguishable string in the comment section. An extra advantage of this labelling is the possibility to conform existing models that could be an important factor for a broader acceptance of reference models.

4.8.2 Product type models

A prime objective of a reference model is to guide the development of more concrete information models that describe a family of similar products. Because of their limited scope, they are able to contain more semantics of this particular set of products. This kind of models will be referenced as *product type* models, in contrast to a product model that contains the data of an actual product.

A product type model should introduce just enough constraints to be meaningful, on the other hand not too many to restrict unnecessarily the domain of products that can be described. In a hierarchy of product type models this balance may vary depending the intended scope of the model, i.e. few constraints at the top level and many constraints at the bottom level.

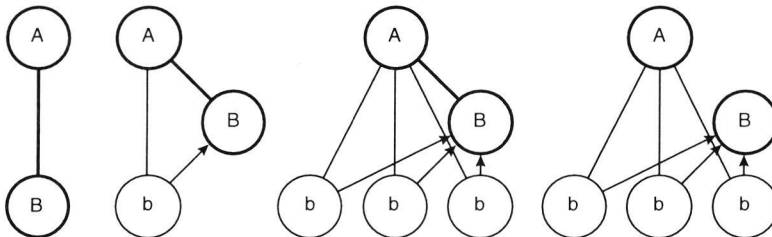


Figure 4.14. Example decomposition configurations with increasing cardinality constraints. Simple edges are decomposes-into relations, while arrowhead edges denote is-occurrence-of relationships.

The specification dimension offers facilities to compose a scale of increasing constraints. An important construct to constrain is decomposition: only a limited domain of possible concepts may occur in a particular decomposition.

Figure 4.14 shows a few configurations:

- first the most general, concept type A decomposes into concept type B. This statement only limits the allowed type of the decomposed concept objects,
- the second case stresses that at least one decomposed concept object is of type B,
- the third case denotes that three or more decomposed concept objects are of type B,
- finally the last case denotes that exactly three decomposed concept objects are of type B.

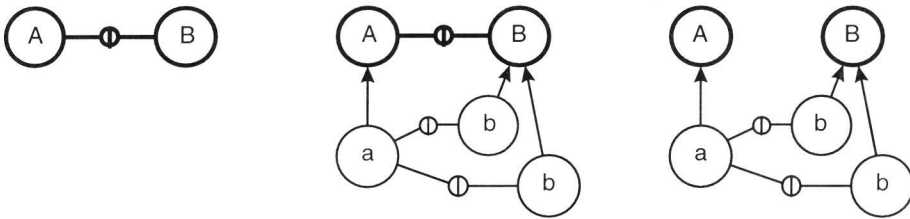


Figure 4.15. Example connectivity configurations with increasing cardinality constraints.

Similar configurations can be formed in the connectivity structure (see figure 4.15):

- occurrences of concept type A can be connected with occurrences of concept type B,
- at least one occurrence of concept type A is connected to at least two occurrences of concept type B,
- exactly one occurrence of concept type A is connected to exactly two occurrences of concept type B.

Figure 4.16 illustrates the top decomposition layer of a simple product type model describing the viaduct product family. The model defines a viaduct to consist of precisely one deck and precisely two abutments both connected with the deck. The product model that can be instantiated from this product type model may also contain zero, one or more pillars. If they are present then they must be connected with the deck³⁹.

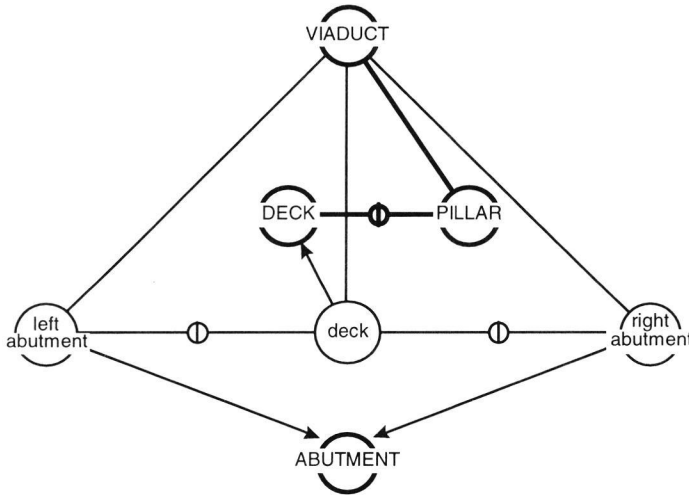


Figure 4.16 A simple product type model of a viaduct.

4.9 Conclusions

Product (and process) modelling is only feasible if the applied models conform to a well defined, flexible and semantically rich conceptual structure. Structuring mechanisms as scope, specification level, decomposition, concretisation, modularity, connectivity and life cycle stage are necessary ingredients for such a super-structure. On the other hand this super-structure should not impose unnecessary constraints on the modelling freedom of product type model developers or force them to use mechanisms they are not interested in. Of course, this last observation is also a very important prerequisite for a

³⁹ Because there is only one deck the connection in the product type model could also be established between the deck occurrence object and the pillar type object. There is no objection to this mixed connectivity in a product type model because the semantics are different. In a product model only actual connectivity is recorded, in a product type model also possible connectivity is stored.

product and process reference model to be accepted by the modelling community.

A flexible product model structure is also a necessity for proposing the appropriate slots for plugging in the various shape models that can be attached to the same product model. If the product type model is well structured each shape model must have a unique slot using its 'co-ordinates' in several (or maybe) all available dimensions. Modularity in shape modelling may help to break down a complete but large shape model (which frequently will address probably several slots) and distribute it in a modular form so that modules and slots form one-to-one relationships.

The next chapter will discuss a shape model representation that is particularly suitable to comply with most of the structuring mechanisms as discussed in this chapter.

4.10 Research Questions

At the end of chapter 3 it was concluded that for product modelling to fulfil its promise (i.e. an application independent data structure that is able to store and retrieve all the information of a certain product during its complete life cycle) it will be obvious that especially the structuring part turns out to be crucial. As a result two more detailed research questions were formulated.

1.1 *Which structuring mechanisms can be employed to organise a product (and process) model?*

The set of structuring mechanisms discussed in this chapter can be encountered in various combinations and interpretations in numerous projects that have addressed this research field. Without claiming completeness this set of structuring mechanisms should belong to the core of available product and process modelling features to fulfil the objectives for product data structures as mentioned in section 4.2.

1.2 *How can these structuring mechanisms be integrated in one super-structure?*

The example product and process model (figure 4.12) shows the advantages of combining various structuring mechanisms in orthogonal modelling dimensions. Thus, a rich set of latent modelling combinations can be spanned leaving

the model developer enough freedom to choose his distinct mixture of modelling constructs.

In chapter 2 research question 2 is formulated as:

2. *Which shape representation is able to meet the requirements of the construction industry?*

This chapter has selected a set of modelling structures to support the development and usage of flexible product models. Data structures for shape modelling are typically developed to satisfy different requirements and usually do not display this flexibility. For shape models to integrate seamless in this modelling environment additional requirements must be stated, especially with respect to modularity. This leads to the following research question:

- 2.1 *Which shape representation is able to meet the necessary requirements with respect to flexibility (modularity) to integrate well with the modelling structures of a product model?*

5

Shape Modelling

Shape representation is traditionally the domain of geometric modelling and computer graphics. Over the years the data structures in geometric modelling have evolved to fulfil requirements in the area of optimal evaluation speed or robust geometric operations support. Product modelling imposes different structural needs, which cannot always be satisfied by the present day shape representation methods. This chapter introduces a new shape representation that is developed from the product modelling perspective: the Space Deformation Tree (SDT) representation. The Space Deformation Tree representation has several characteristics that make it particularly suitable for association with product models.

5.1 Introduction

This chapter describes a method to represent the shape of an object. In the past numerous shape representation schemes have been developed for all kinds of purposes (mostly with realistic and/or fast visualisation in mind). The representation method that will be discussed here is selected because its structure is remarkably isomorphic to the kind of data structures that are often encountered in product modelling. Besides this property, this representation method, especially the topological part, is developed with generalisation rather than implementation efficiency as the prime objective. The geometrical part has some built-in features that makes it particularly fit for long stretched objects often found in civil engineering like highways, railroads, bridges, tunnels, dikes, etc. Almost all building blocks of this method can be found in other representation schemes, nevertheless the composition and especially the geometric part is original and a result of this PhD research. For reasons that will be explained in the remainder of this chapter the method is called Space Deformation Tree (SDT) representation.

The Space Deformation Tree representation deals with three modelling areas that are discussed in separate sections

- The first section considers geometry as the modelling of unbounded spaces:

- its basic structure defined by the dimensionality,
- the embedding relations between spaces through location, orientation and deformation, and
- operations on spaces like inclusion and integration.
- The second section considers topology the modelling of bounded portions of space cells:
 - its basic structure defined by the dimensionality,
 - the enclosing and bounding relations between cells,
 - the interface between cells and spaces, and
 - operations on cells like inclusion, exclusion and integration.
- The third section integrates the former two sections on the actual level of shape modelling:
 - the building stones: shape primitives,
 - the assembly object: shape complex, and
 - the relational object: shape occurrence.

5.2 Space Modelling

Shape representation is in essence an allocation problem of modelling space: which part of space is claimed for what. The first choice should be the dimensionality of the modelling space: a two-dimensional shape needs at least a two-dimensional modelling space; a one-dimensional space cannot contain a three-dimensional shape. One global modelling space is sufficient to represent any shape, still introducing many interrelated modelling spaces obtains much more flexibility. This section considers the use of multiple spaces of different dimensionality in a tree-like structure as the flexible geometry foundation of the SDT-representation.

5.2.1 Dimensionality

Mathematically a modelling space can be defined as a (generally infinite) set of points. A point represents an infinite small portion of space. A set is an unordered aggregation structure therefore point sets are normally declared isomorphic to real number sets. For example, a one-dimensional space is a point set that is isomorphic to the real number set \mathbb{R} . Because the real number set \mathbb{R} has a total ordering, all points in the point set are also ordered. As a result:

- each point in a one-dimensional space can be addressed using a real number,

- two points can be compared by comparing their real number addresses, and
- the distance between two points can be calculated by a simple subtraction.

A special point, called the origin, is addressed with 0. All other point addresses can be interpreted as the distance (or better off-set) of that point to the space origin.

A one-dimensional space has only one intrinsic direction: forward (direction positive infinite) or its antipode backward (direction negative infinite). Higher order modelling spaces have more orthogonal directions and hence, are isomorphic to a product set of real number sets: \mathbb{R}^n . Where n stands for the dimensional order:

- $n = 1$: one-dimensional space: isomorphic to \mathbb{R}

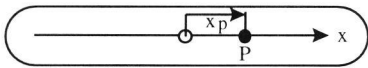


Figure 5.1 One-dimensional space.

- $n = 2$: two-dimensional space: isomorphic to $\mathbb{R}^2 = (\mathbb{R} \times \mathbb{R})$

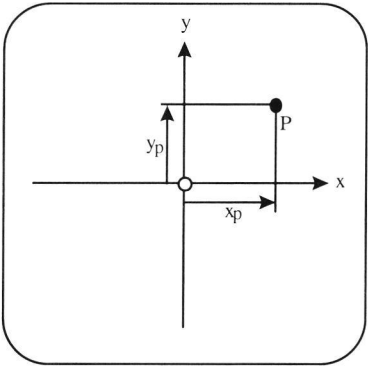


Figure 5.2 Two-dimensional space.

- $n = 3$: three-dimensional space: isomorphic to $\mathbb{R}^3 = (\mathbb{R} \times \mathbb{R} \times \mathbb{R})$

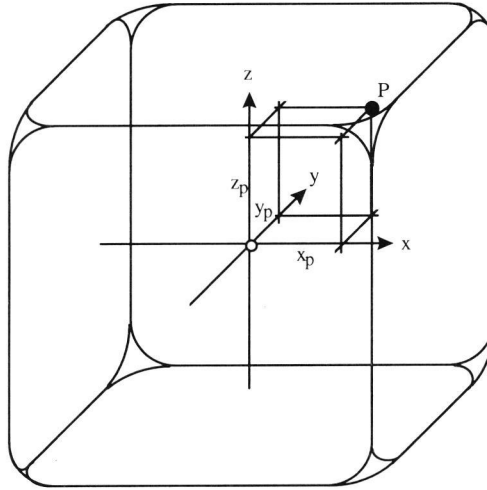


Figure 5.3 Three-dimensional space.

n defines the number of orthogonal directions and the size of the point co-ordinates tuple. A co-ordinates tuple contains two ordered real numbers for a point in a two-dimensional space, three ordered real numbers for a point in a three-dimensional space, etc. Normally a Cartesian co-ordinate system will be applied but there is no objection using a polar co-ordinate system in a two-dimensional space and cylindrical or spherical co-ordinate systems in a three-dimensional space.

A special case is the zero-dimensional space. It contains only one singular point, by definition its origin, that need no addressing system or intrinsic direction.

Shapes in normal Euclidean space can always be modelled in three dimensions. Spaces with dimensionality 4 (or even more) may use this extra dimension for special purposes like a temporal component or perspective transformations. Moreover, a three-dimensional space can also be used to model for example two-dimensional shapes and their behaviour in time.

5.2.2 Embedding

As mentioned before, introducing more than one modelling space offers much more flexibility.⁴⁰ For example, a library object is defined in a local mod-

⁴⁰ A similar approach can be found Van Emmerik's tree of co-ordinate systems [Emmerik 1990].

elling space then (after copying it) the modelling space of the application model will embed this local space including the library object. All locally defined co-ordinates of the library object are unaffected.

In general a modelling space of dimensionality n may embed other modelling spaces of dimensionality m , where $m \leq n$. If $m < n$ then the point set of the embedded space is a genuine subset of the point set of the embedding space. If $m = n$ the point set of the embedded space is identical to the point set of the embedding space.

The *embedding/embedded_by* relationship between two modelling spaces can be split in three components: a location component, an orientation component and a deformation component.

- *location*

The location component holds the location of the origin of the embedded space in the co-ordinate system of the embedding space. For a rigid *embedding/embedded_by* relationship (i.e. no translation degrees of freedom) the location component should always be present.

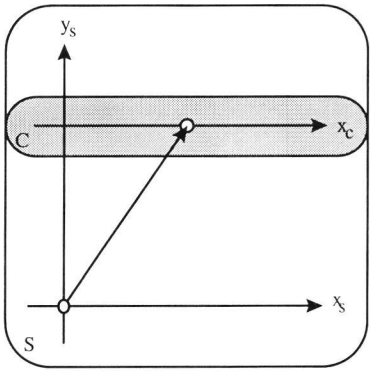


Figure 5.4 A two-dimensional space S embeds and locates a one-dimensional space C .

- *orientation*

The orientation component holds the orientation of the embedded space relative to the orientation of the embedding system. Orientation in a three-dimensional space can be specified with two so-called orientation vectors each indicating the orientation of a pre-defined direction in the embedded space. This definition indicates that the orientation component can be neglected if the embedded space is a zero-dimensional space (no intrinsic directions). If the embedded space is a one-dimensional space only one one-dimensional orientation vector will be sufficient (one intrinsic direction). If the embedding space is two-dimensional only one two-dimensional orientation vector

will be sufficient. At last, a one-dimensional embedding space may toggle between forward and backward orientation of the embedded one-dimensional space.

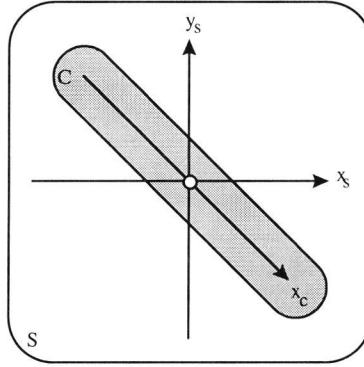


Figure 5.5 A two-dimensional space S embeds and orientates a one-dimensional space C .

It is common practice to choose the primary orientation along the first co-ordinate axis and the secondary axis along the vector product of the first and the second co-ordinate axes. The secondary orientation coincides with the normal vector of a two-dimensional space (surface) at its origin.

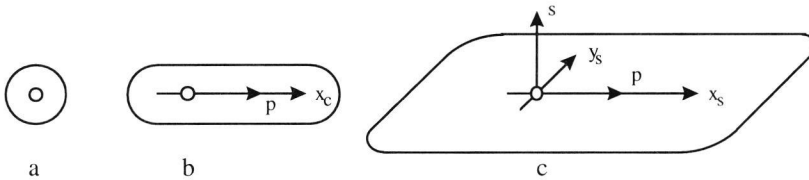


Figure 5.6 Orientations for (a) a zero-dimensional space: none, (b) a one-dimensional space: coincides with the only available co-ordinate axis and (c) a two dimensional space: coincides with the first co-ordinate axis and the normal vector in the origin.

- *deformation*

Straight and flat geometry is adequately served with the location and orientation components of the *embedding/embedded_by* relationship between spaces. The deformation component is needed to mould the straight and flat spaces into curved spaces. The deformation component is specified by a mapping function that returns a point in the point set of the embedding space for each point from the point set of the embedded space. Additional requirements are necessary to exclude ‘wild’ mapping functions that blow up a space after deformation. Basically continuity should be preserved after de-

formation: neighbour points before deformation should still be neighbour points after deformation, or more formally:

The mapping function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with $3 \geq m \geq n \geq 0$ is continuous if for each $\underline{p} \in \mathbb{R}^n$ is asserted that if for each $\epsilon > 0$ a $\delta > 0$ exists, such that for all $\underline{x} \in \mathbb{R}^n$ with $|\underline{x} - \underline{p}| < \delta$ applies $|f(\underline{x}) - f(\underline{p})| < \epsilon$.

If the mapping function has an inverse function, at least valid for the region in the modelling space that contains shapes, the possibility is open to relate points in the deformed space to their original location in the undeformed space.

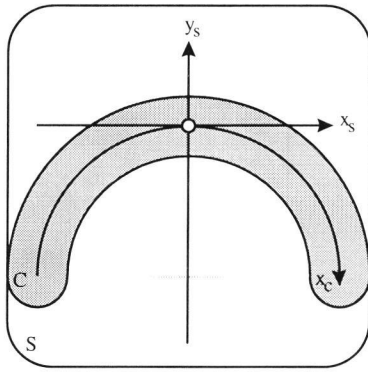


Figure 5.7 A two-dimensional space S embeds and deforms a one-dimensional space C .

Mapping functions may be implemented as analytical functions or as numerical integration algorithms or simply as a table look-up procedure from earlier calculations results.

The deformation component in the *embedding/embedded_by* relationship offers great opportunities to build up reusable components. A component, that is originally defined in local straight and flat geometry, can be moulded in the desired shape after its incorporation in the current modelling space context.

5.2.3 Symbolic notation

Figure 5.8 displays the diagram symbols that will be applied throughout the remainder of this chapter. The number of orthogonal directions symbolises the dimensionality of the space. The identifier is assembled from a prefix character and a serial number:

- zero-dimensional space: P (Point),
- one-dimensional space: C (Curve),
- two-dimensional space: S (Surface),
- three-dimensional space: U (Universe).

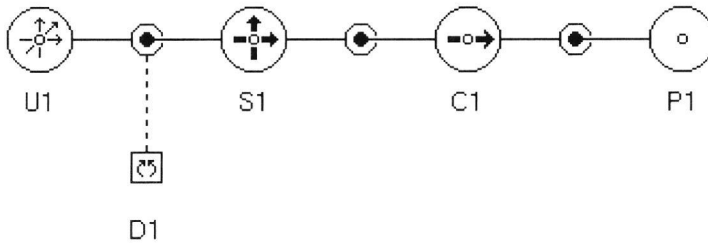


Figure 5.8 Diagram symbols representing a space graph.

The sense of the relations is from embedding space to embedded space. Only the deformation component (small square) of the embedding relation is shown to keep the diagrams as elementary as possible.

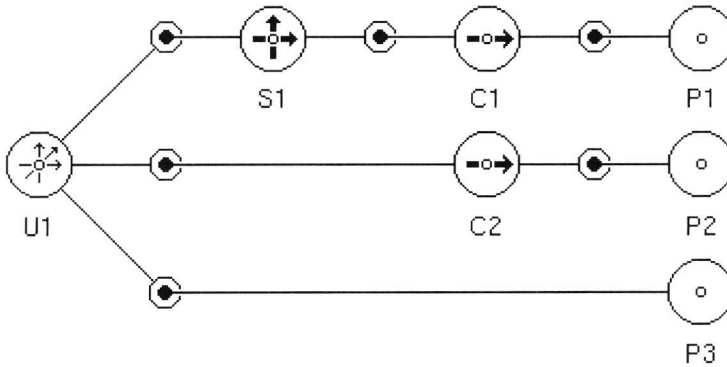


Figure 5.9 Example of a space graph.

Figure 5.9 shows an example of a space graph. A space graph is a graph with nodes representing spaces and links representing *embedding/embedded by* relations. The space graph is a so called directed a-cyclic graph (DAG), which means that the links are directed (from source node to target node) and that traversing any path according its directional sense, a node is never visited more than once. Usually a space graph will have a single root space, here the three-dimensional space U1. This root space embeds three subspaces: a two-dimensional space (surface) S1, a one-dimensional space (curve) C2 and a zero-dimensional space (point) P3. Another one-dimensional space (curve) C1 is embedded in S1 and thus indirectly in U1. It may be possible that C2 turns out to be also a sub-

space of S1 but C1 is *guaranteed* a subspace of S1. This assertion will remain true as long as the S1/C1 relation is in place, hence modifications in location, orientation or deformation components will not change this status.

5.2.4 Space graph composition

Composing a space graph can be done in several ways. One option is to create a root space and then to let it grow in a tree-like manner until the leaves (usually zero-dimensional spaces: the points). As long as the graph structure is a pure tree (e.g. starting from the root space each space node can only be reached traversing a unique path) the space structure will always be consistent. The reason is that a tree-structure is a non-redundant data structure. Each data item is specified only once and cannot be obtained as the result of computations on other stored data items.

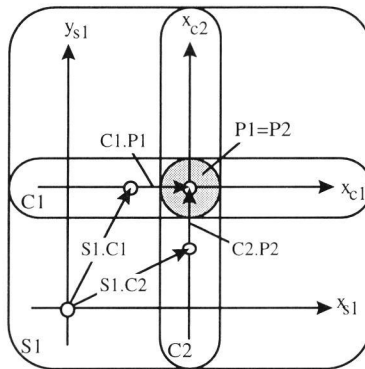


Figure 5.10 Example of two one-dimensional spaces each referring to a separate zero-dimensional space. The zero-dimensional spaces refer to the same point (set).

Another option is to start from the leaves and bundle them into superspaces until the root space is composed. It is also possible to plug a complete space graph with its root space into a space node of another space graph. This last situation is often the case when including a predefined library object in a current model.

After composition it is not unusual that one or more spaces refer to an identical point set. The example, shown in figure 5.10 and 5.11, exhibits two one-dimensional spaces in a two-dimensional space. Each one-dimensional space embeds a zero-dimensional space, that actually refer to the same point.

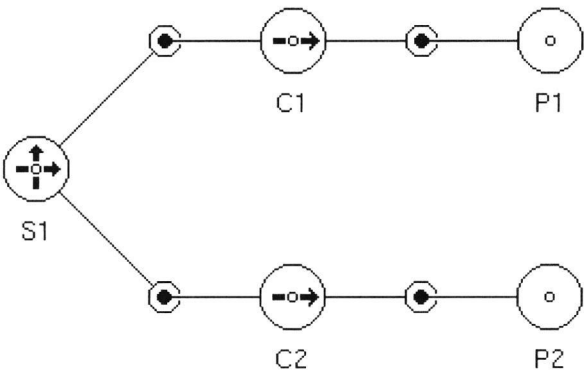


Figure 5.11 The same example represented as a space graph diagram.

It is because C1 and C2 are embedded in the same super-space S1 that such a conclusion can be drawn. S1 is a common reference frame for comparison. In many applications it is essential to detect these identical point sets and therefore shared spaces. An operation called space integration will spot these identical

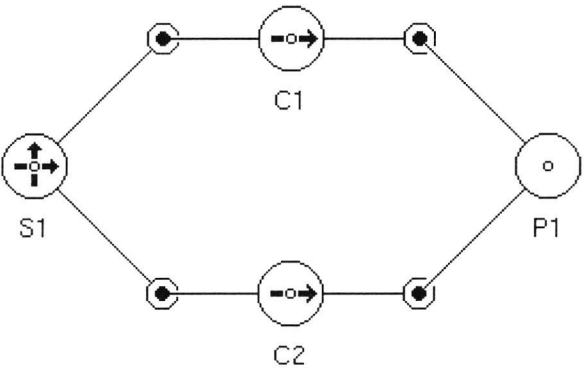


Figure 5.12 The same example after space integration.

point sets and will rewire the space graph to make these shared spaces explicit.

Figure 5.12 displays the effect of such a space integration operation. The result will never be a space tree anymore therefore the structure must be frozen to keep it consistent. Before editing such a structure it should be melted first by converting it into a space tree. Then the embedding relation components can be modified after which a new space integration operation is executed.

A possible algorithm to implement the space integration operation is listed below in pseudo code. The code is simplified by assuming no deformation components in the space relations.

The operation is distributed over three passes.

- The first pass simply removes all embedded spaces with equal dimensionality as its embedding space.
- The second pass searches for identical point sets, rewires the relations to one space and deletes the other space.
- Finally, the third pass searches for extra embedding relations between spaces that were not, directly or indirectly, connected yet.

```
procedure integrate;
  mark all subspaces unvisited;
  root_space.integrate1;
  mark all subspaces unvisited;
  root_space.integrate2;
  mark all subspaces unvisited;
  root_space.integrate3;
end; -- integrate

procedure integrate1;
  for all unvisited subspaces do
    begin
      if this.dimensionality = subspace.dimensionality
        ... eat subspace ...
      else
        subspace.integrate1;
      end;
      mark visited;
    end;
  end; -- integrate1

procedure integrate2;
  for all unvisited subspaces do
    begin
      for all other spaces do
        begin
          if subspace.dimensionality = other_space.dimensionality
            begin
              compare (subspace, other_space);
              if subspace.point_set = other_space.point_set
                begin
                  rewire combined relations to subspace;
                  delete other_space;
                end;
            end;
          end;
        end;
      end;
      subspace.integrate2;
    end;
    mark visited;
  end; -- integrate2

procedure integrate3;
  for all unvisited subspaces do
    begin
      for all non-parent superspaces do
        begin
          compare (subspace, superspace);
          if subspace.point_set  $\supseteq$  superspace.point_set
            add new embedding relation;
          end;
        end;
      end;
      subspace.integrate3;
    end;
    mark visited;
  end; -- integrate3
```

The operation to convert a structure, that was the result of a space integration operation, into a pure space tree simply duplicates all spaces with more than one *embedded_by* relationship and rewires accordingly.

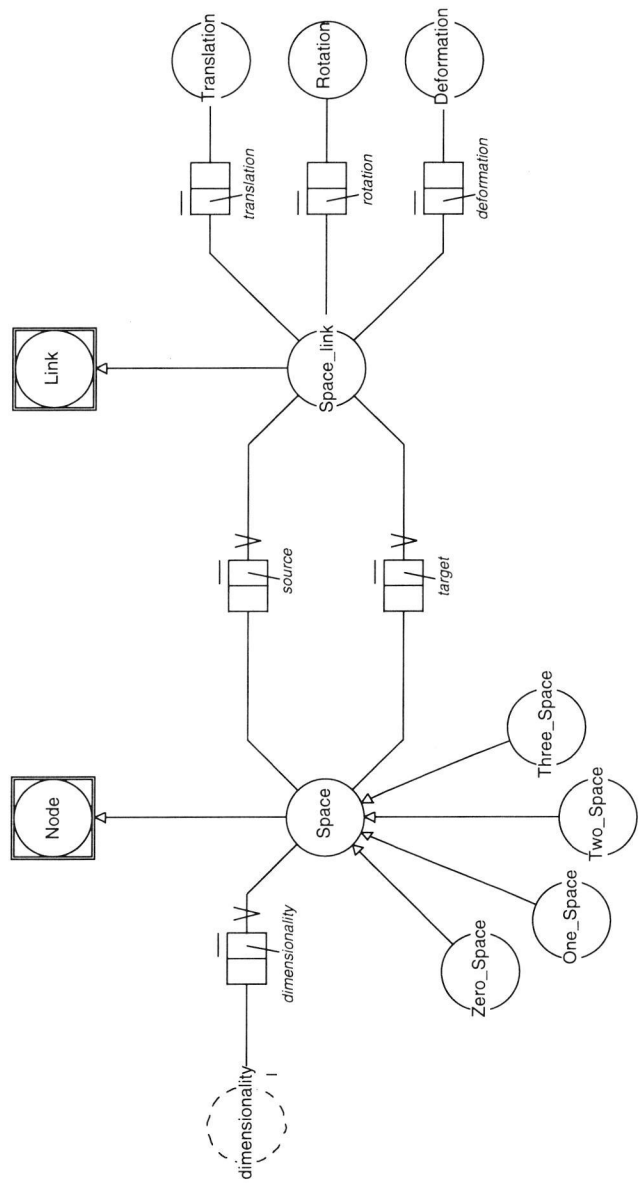


Figure 5.13 Data model for spaces.

5.2.5 Spaces data model

A data model for spaces can easily be founded on a data model for graphs. This graph data model should represent arcs (links, edges) as explicit objects, because the space structure attaches specific data to the *embeds/is_embedded_by* relation, i.e. the translation, rotation and deformation components. A valid space structure conforms to a special type of graph: the directed a-cyclic graph. However, to facilitate the creation and editing of space graphs the data model should not impose too many restrictions to force a legitimate data structure. A verification routine that can be invoked at a moment of a stable space structure offers much more flexibility. Figure 5.13 shows a NIAM data model that could act as the backbone data structure to accomplish space modelling. The explicit entity types for zero-, one-, two- and three-dimensional spaces are strictly not necessary, but in an object-oriented system these descendent classes offer the occasion to add specialised routines.

5.3 Cell Modelling

In the introduction of the previous section, shape representation is characterised as in essence an allocation problem of modelling spaces. In other words which part of space is claimed for what. Succeeding the discussion of the concept of modelling space itself this section considers the use of cells, i.e. bounded portions of modelling space, in a graph structure as the topological basis of the SDT-representation. With a ‘bounded portion of modelling space’ is meant one piece of contiguous space without singular points or holes.

The next sections will discuss classification of cells by:

- their dimensional order,
- the relationships between cells (which are partly mandatory for a correct cell specification),
- the interface between cells and spaces, and finally,
- the composition of cells.

5.3.1 Dimensionality

Similar to modelling spaces modelling cells can be categorised according to the dimensionality of the space they refer to:

- a zero-dimensional cell refers to exactly one point.
- a one-dimensional cell refers to a portion of a one-dimensional space.

- a two-dimensional cell refers to a portion of a two-dimensional space.
- a three-dimensional cell refers to a portion of a three-dimensional space.

These definitions are generally valid. Nevertheless depending the structure of the space graph, lower dimensional cells of course also enclose portions of the embedding super-spaces. However, their primary reference is to a space with the same dimensionality.

The point set of a cell can be decomposed into two disjunct point subsets: a set of interior points and a set of boundary points. An interior point of a cell has an environment of neighbouring points in all directions of the space it resides in while a boundary point has not such a total environment. Formally stated for the point set **P** of a cell:

$$\mathbf{P} = \mathbf{P}_{\text{interior}} \cup \mathbf{P}_{\text{boundary}} \text{ and } \mathbf{P}_{\text{interior}} \cap \mathbf{P}_{\text{boundary}} = \emptyset$$

Figure 5.14 illustrates both sets for a one-dimensional cell. The boundary point set of a one-dimensional cell always contains exactly two different boundary points.

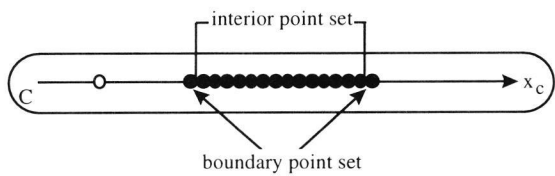


Figure 5.14 Interior point set and boundary point set of a one-dimensional cell in a one-dimensional space C.

5.3.2 Symbolic notation

In addition to the graphical symbols that were previously defined to represent spaces the next set of symbols will be used to represent cells:



zero-dimensional cell: V (Vertex),



one-dimensional cell: E (Edge),



two-dimensional cell: F (Face),




three-dimensional cell: B (Body).

The cell id is assembled from a cell type character and a unique serial number for that cell type. The cell type character refers to common topological entity names, however, keep in mind that the cell definitions are slightly different from their common topological counterparts.

5.3.3 Enclosing and Bounding

If a modelling cell refers to a portion of modelling space it could easily happen that an overlapping piece of space is claimed by another cell. If this piece of space falls completely in the domain of the first cell (i.e. the point set of the second cell is a subset of the point set of the first cell), the first cell is said to enclose the second cell. Equivalently, the second cell is enclosed by the first cell.

Cell a is enclosed by cell b if their point sets relate as: $P_a \subseteq P_b$

The enclosure relation is represented by: . Connectors are attached to either an outer contour or an interior area to depict the direction of the relation. This is the basic relation type cells in a modelling cell graph may declare. A necessary requirement for an enclosure relation is the allowed difference of dimensionality between an enclosing cell a and an is_enclosed_cell b :

$$\dim(a) \geq \dim(b)$$

Which means that a cell cannot enclose another cell that has a dimensionality greater than itself. Figure 5.15 illustrates an enclosure relation between a one-dimensional cell E1, that allocates a portion of a one-dimensional space C1, and a zero-dimensional cell V1, that allocates a zero-dimensional space P1. P1

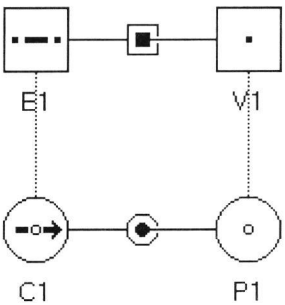


Figure 5.15 *Encloses/is_enclosed_by* relationship between a one-dimensional cell E1 (carried by a one-dimensional space C1) and a zero-dimensional cell V1 (carried by a zero-dimensional space P1).

is a subspace of C1 and thus it may be possible that V1 is enclosed by E1.

However, to answer this question unambiguously it must be clear which part of C1 is allocated by E1. This is only feasible by specifying a closed boundary

around the interior space of the cell. Of course, this boundary is a bounded portion of space itself, and hence can be specified using another cell or several other cells. This idea of defining cells with other cells should lead to a paradox, but fortunately the boundary space of a n -dimensional cell has a dimensionality of $n - 1$. This sequence comes to a halt by a zero-dimensional cell, because a zero-dimensional cell contains a single point and therefore is it sufficiently specified in itself. A zero-dimensional cell has a $\mathbf{P}_{\text{interior}}$ with one element and an empty $\mathbf{P}_{\text{boundary}}$. As a result, defining cells with other cells propagates a correct recursive definition with the zero-dimensional cell as a proper terminating definition.

In addition to the *encloses/is_enclosed_by* relation type a *bounds/is_bounded_by* relation type is needed to specify all cells with dimensionality > 0 .

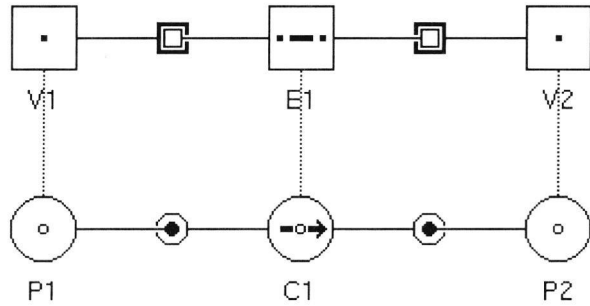



Figure 5.16 Specification of a one-dimensional cell bounded by two zero-dimensional cells.

Figure 5.16 displays the necessary structure to define a one-dimensional cell bounded by two zero-dimensional cells.

Boundary relations build up the boundary point subset $\mathbf{P}_{\text{boundary}}$ of a cell. The next relation holds for any cell bounded by n other cells:

$$\mathbf{P}_{\text{boundary}} = \mathbf{P}_{1,\text{interior}} \cup \dots \cup \mathbf{P}_{i,\text{interior}} \cup \dots \cup \mathbf{P}_{n-1,\text{interior}} \cup \mathbf{P}_{n,\text{interior}}$$

The *bounds/is_bounded_by* relation type (symbolised by: ) can be interpreted as a special case of the *encloses/is_enclosed_by* relation type. An enclosure relation refers to a point subset that may include both interior points as well as boundary points, a boundary relation refers, by definition, to a point subset that includes only boundary points. A necessary requirement for a boundary relation is the difference of dimensionality between the bounding cell a and the *is_bounded_by* cell b :

$$\dim(a) = \dim(b) - 1$$

A cell boundary that is defined with one or more boundary cells should constitute a closed, non-intersecting and non-redundant or overlapping structure. Closed means that starting from a point inside the cell and traversing, along any

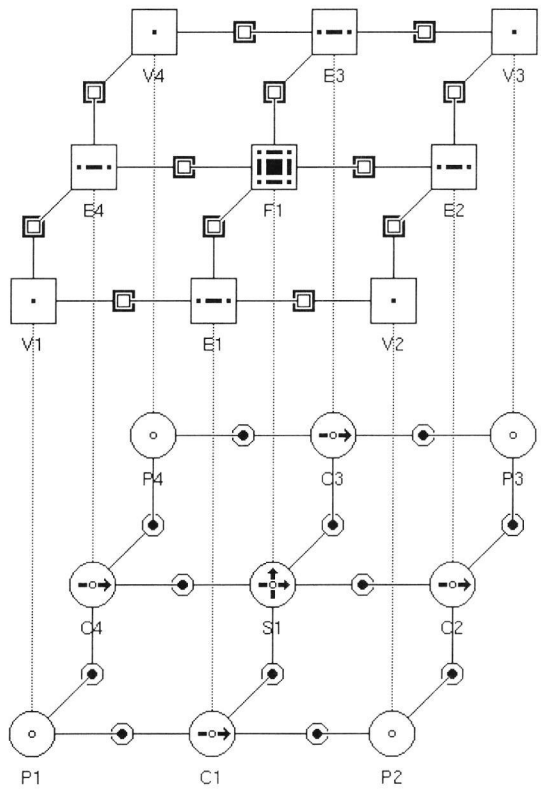


Figure 5.17 Specification of a two-dimensional cell bounded by four one-dimensional cells.

path, to a point outside the cell always a point of the boundary should be encountered. A non-intersecting and non-redundant or overlapping structure means that all point sets of the participating boundary cells are disjunct.

For the example of figure 5.16 this implies that a closed boundary for a one-dimensional cell can be assembled from precisely two non-identical boundary points (referenced by two zero-dimensional cells). Figure 5.17 shows another example of a two-dimensional cell bounded by four one-dimensional cells.

5.3.4 The interface between cells and spaces

The relationship between spaces and cells is of cardinality one-to-many: a cell references only one space, but a space may carry many cells. The cells and spaces interface offers a variety of possibilities to model apparently identical modelling patterns.

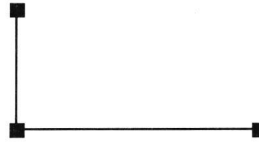


Figure 5.18 Modelling example consisting of two intersecting line segments.

A simple modelling example is shown by figure 5.18. Obviously it consists of two line segments in a L-configuration. The most simple way to model this is by two separate one-dimensional cells and a space graph with two separate branches as is illustrated in figure 5.19.

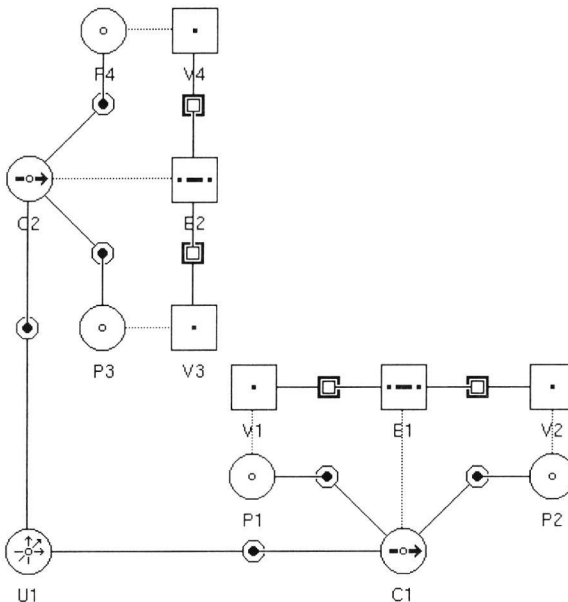


Figure 5.19 Two independent line segments.

This configuration is very flexible (lots of freedom to vary location parameters) nevertheless some applications have to know if the two line segments are connected or not. This knowledge should be available in an explicit form not

via geometrical calculations with a limited reliability. This connection can be specified on two levels: a geometrical connection or a topological connection (which implies also a geometrical connection). Figure 5.20 displays the configuration with an explicit geometrical connection.

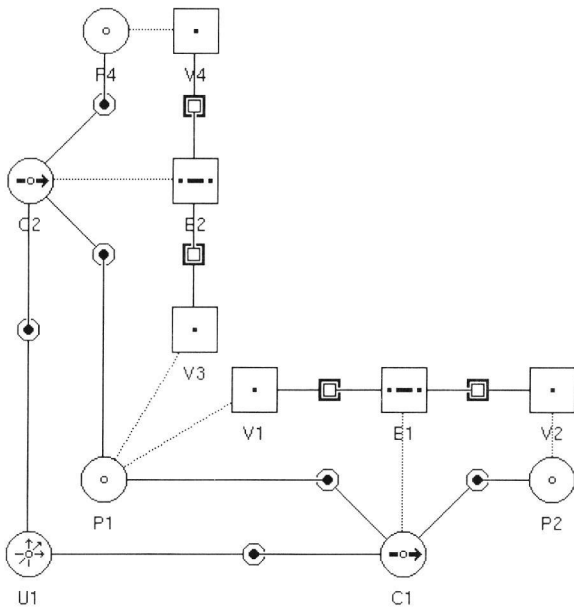


Figure 5.20 Two line segments with shared geometry.

Stating that the zero-dimensional space P1 is embedded in both one-dimensional spaces C1 and C2 makes the geometrical connection explicit. This single zero-dimensional space is now the carrier of both zero-dimensional cells V1 and V3. Finally figure 5.21 displays the configuration with an explicit topological connection. Here both one-dimensional cells E1 and E2 share a boundary cell: the zero-dimensional cell V1, hence a topological connection. All three modelling configurations have their specific usage's. From the product modelling view the requirements may vary from a very loose description up to a highly integrated description. SDT can be configured to meet these requirements.

The modelling freedom facilitates also various representations on the implicit/explicit scale. For example, the operands of a Boolean operation can be recorded in advance but also in combination with the result of the operation, while preserving the original geometry. As a result the fragments still refer to the geometry of the operands. The next section will explore a few possibilities in that area.

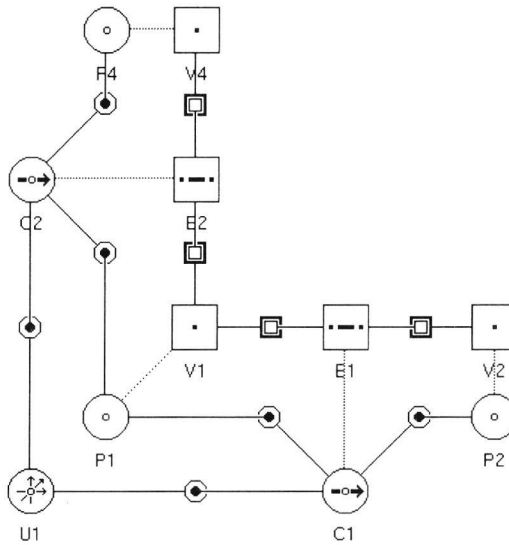


Figure 5.21 Two completely integrated line segments, i.e., with shared geometry and shared topology.

5.3.5 Inclusion and exclusion

A cell is defined as a bounded portion of space without singular points or holes. Nevertheless, there is certainly a need to define holes or voids in faces or solids. Defining the enclosure relationship between cells to be exclusive could realise this, i.e. the point sets of the enclosing cell and the enclosed cell are defined to be disjunct. In a product modelling context this solid/void interpretation can be generalised in two mutually exclusive spatial claims issued by two distinct product definition units. By generalising this principle it becomes clear that there will be also a need for inclusive spatial relationships, i.e. two product definition units that share a spatial claim.

Although in a 'pure' shape model only the exclusive interpretation is meaningful there does not seem to be an evident standard interpretation, exclusive or inclusive, in a product modelling context. Therefore, this interpretation must explicitly be attached to each cell relationship: enclosure relations and boundary relations. This leads to four combinations:

- Exclusive enclosure relation
- Inclusive enclosure relation
- Exclusive boundary relation
- Inclusive boundary relation

The exclusive relationship introduces an additional definition for the point set of a cell: the logical point set: P_{logical} . The point set definition for a cell that is used until now (which will be referred from now on as the physical point set: P_{physical}) consists of two disjunct point sets P_{boundary} and P_{interior} . P_{physical} is one continuous point set, i.e. without voids, singular points, etc. P_{logical} is the point set that results if all direct and indirect enclosure relationships are evaluated and may therefore consist of multiple disconnected point sets with internal voids. If cell a exclusively encloses cell b then P_{logical} can formally be defined as:

$$P_{\text{logical}}(a) = P_{\text{physical}}(a) - P_{\text{logical}}(b)$$

In a product modelling context it is P_{logical} that is intended if a product defi-

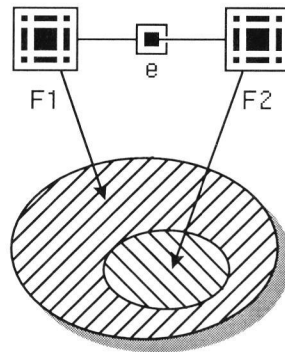


Figure 5.22 Exclusive enclosure relation.

inition unit references a cell for its spatial claims.

The exclusive enclosure relation (figure 5.22) covers the well-known holes or voids structures in common topological data structures, although the exclusive enclosure relation generalises this solid/void concept so that it now comprises:

- a vertex void in a vertex, edge, face or body
- an edge void in an edge, face or body
- a face void in a face or body
- a body void in a body.

It must be stressed that the void interpretation is the standard interpretation in a pure shape model. However, in a product modelling context the enclosed cell could be claimed by a product definition unit that fills up this 'void', e.g. the classical example of a window-in-a-wall.

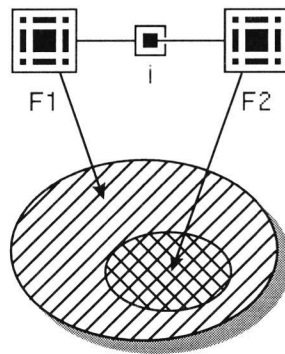


Figure 5.23 Inclusive enclosure relation.

The inclusive enclosure relation (figure 5.23) offers the opportunity to double claim certain areas to attach additional product information. This proves to be especially important in a process modelling context: several activities (concurrent or time sequential) that claim a certain spatial area, e.g. several passes of a milling machine.

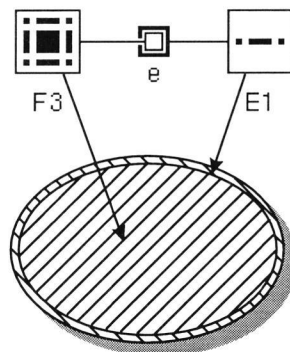


Figure 5.24 Exclusive boundary relation.

The exclusive boundary relation or open boundary relation (figure 5.24) is important if there is a need to distinguish the interior of something and its boundaries. A classical example from the building industry is the distinction between the spatial system of rooms, corridors, lift-shafts and the structural separation system of walls and floors. Exclusive boundary relations are particularly inevitable if it is undesired that certain points are claimed more than once.

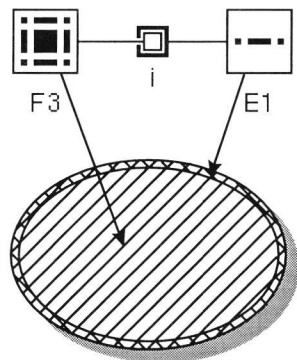


Figure 5.25 Inclusive boundary relation.

The inclusive boundary relation or closed boundary relation (figure 5.25) is applicable if there is no distinction between the interior of something and its boundaries. This will be the normal case for a solid model representation of an object of homogeneous material (without coatings or foils at its perimeter).

The introduction of the inclusive/exclusive predicate for enclosure relations contributes great flexibility in the specification of the $\mathbf{P}_{\text{logical}}$ point sets of cells, and hence flexibility in the specification of the definition areas of product def-

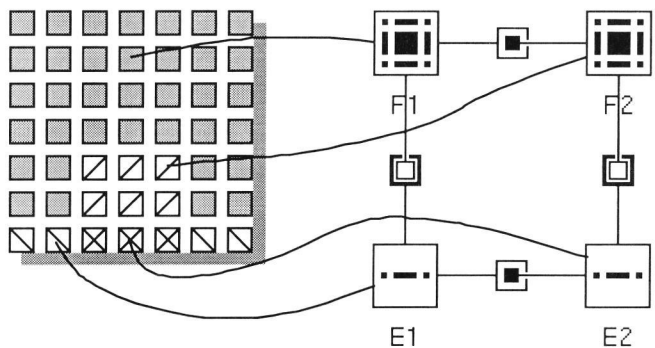


Figure 5.26 Cell graph with undetermined enclosure and boundary relations.

inition units. An exhaustive example is denoted by figure 5.26. It shows a (part of a) cell graph with two- and one-dimensional cells (F1, F2, E1, E2) interconnected by both enclosure and boundary relations. The next table lists sixteen different inclusive/exclusive combinations that can be assembled from these 4 relations.

	F1/F2	F1/E1	E1/E2	F2/E2
0	i	i	i	i
1	i	i	i	e
2	i	i	e	i
3	i	i	e	e
4	i	e	i	i
5	i	e	i	e
6	i	e	e	i
7	i	e	e	e
8	e	i	i	i
9	e	i	i	e
10	e	i	e	i
11	e	i	e	e
12	e	e	i	i
13	e	e	i	e
14	e	e	e	i
15	e	e	e	e

The resulting P_{logical} (F1) point sets for all these sixteen cases are presented

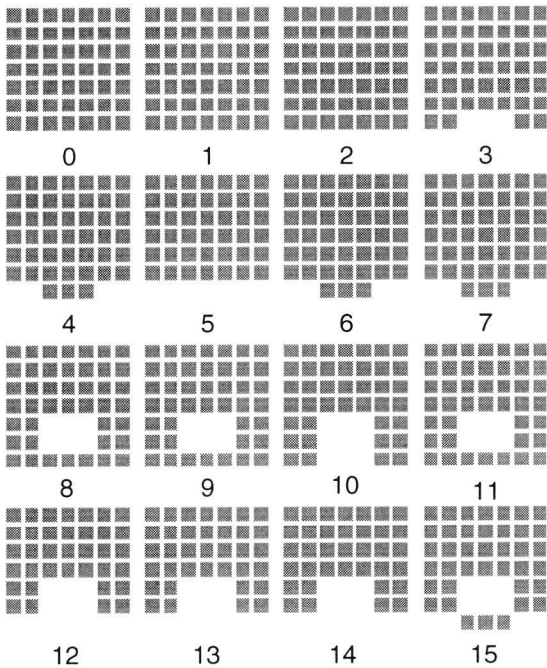


Figure 5.27 P_{logical} (F1) for all sixteen possible combinations.
in the next figure.

Certain combinations lead to an identical result, however, the cases will also deviate in the $\mathbf{P}_{\text{logical}}$ point sets of the other participating cells. This influences the distribution (multiple usage) of points in the various $\mathbf{P}_{\text{logical}}$ point sets.

5.3.6 Cell graph composition

The composition of cells in a cell graph differs from the composition of spaces in a space graph. Initially, a space graph is a tree that can be augmented by additional embedding relations, making explicit that was already present implicitly. A cell graph is mostly a forest, i.e. multiple root cells that refer to dimensional lower order cells until the zero-dimensional cells (vertices) as leaf cells are reached. Initially a cell forest can be composed from a set of disconnected cell trees. Through an integration operation these cell islands can be connected, changing the structure into a multiple entry directed a-cyclic graph. The example discussed in section 5.3.4. shows the general approach:

- A pure space tree carries the various isolated cell trees (figure 5.19). All cell/space relations are of type one-to-one.
- After a space integration operation, the space tree has changed into a single entry directed a-cyclic graph. The cell graph is not affected however certain cells are now carried by the same space.
- Finally, a cell integration operation changes the forest of disconnected cell trees into a multiple entry directed a-cyclic graph. Some of the cells that share the same carrier space will fuse into a single cell, restoring the one-to-one relationship between cell and space.

The cell integration operation will primarily investigate the cells that share a common space to decide for cell fusion or the addition of an enclosure relation.

5.3.7 Cells data model

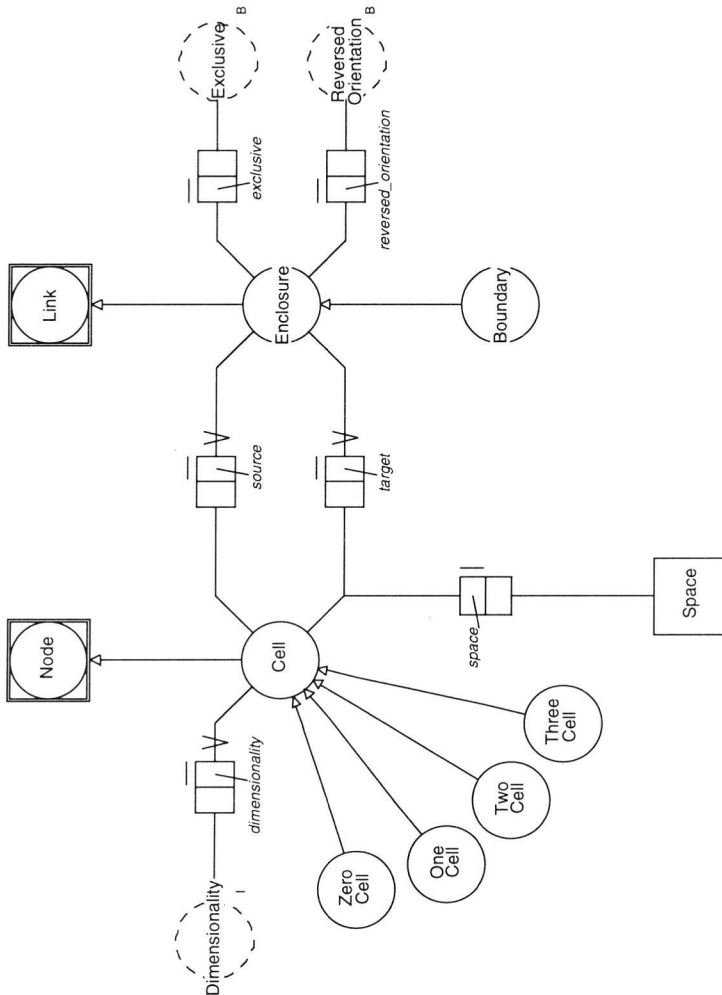


Figure 5.28 Data model for cells.

Analogous to the spaces data model a cells data model can easily be based on a graph model with explicit link objects. Although the explicit link objects are less urgent here, the definite many-to-many relationship between cells is more manageable with such an entity. The data structure denoted in figure 5.28 makes the subtype relation between enclosure and boundary explicit. Additional rules will be necessary to force the closed boundary point set and the point sub-set character of the enclosure relation.

Two Boolean attributes are defined for the enclosure/boundary entity:

- the inclusive/exclusive switch is the predicate that sets the union or difference interpretation of the relation.
- the `reverse_orientation` switch is used by applications that apply specific orientations for certain cells (mostly edges or faces). For example, solid modelling applications often define the normal vector of a face to point outside the material, while other applications force a right hand rule to the loop of edges that form the contour of a face (and a left hand rule for the holes in a face).

Normally cells copy their orientation from the space they refer to, hence the `reverse_orientation` switch can be used to toggle this option.

5.4 Shape Modelling

In the former two sections two graph structures are defined to model spaces (geometry) and cells (topology). In general this description level is too low for a direct interaction with product modelling. This section raises the aggregation level to complete combinations of space graphs and cell graphs to define an axiomatic shape primitive set. This axiomatic shape primitive set can be extended with a shape primitive set that can be derived from the axiomatic set to form a set of shape primitives that are normally implemented in general purpose CAD-systems.

Shape primitives can freely be aggregated in a so called shape complex. Recursively, a shape complex may again aggregate other shape complexes to form again a directed a-cyclic graph.

Finally, shape occurrence objects are discussed to reduce shape structures with much repetition and to help implement type/occurrence decomposition.

5.4.1 Shape Primitive





A shape primitive is composed from a well-formed combination of a space graph and a cell graph and that will act as a leaf node in a higher level graph type: the shape graph. Shifting to a higher level of shape modelling reduces considerably the number of primary node objects involved to the expense of modelling freedom. The object reduction will turn out an important advantage because it offers in principle a one-to-one relationship between product definition units and shape representations. The modelling freedom is primarily restricted by the reduction of possible cell configurations to model similar shapes,

as a consequence the modelling freedom at shape graph level is not really affected.

Two approaches can be followed to define a shape primitive set. First a more mathematical approach to limit the set to a bare minimum of shape primitives that are independent from each other. Hence, no primitive shape can be defined from one or more other primitive shapes. On the other hand the set must be complete to act as building blocks to compose any shape in principle. This will be called the axiomatic shape primitive set. Another approach is to provide a primitive set that is large enough to fulfil a majority of shape requests immediately. Clearly the extra shape primitives in this approach can be derived from the axiomatic shape primitive set and hence will be called the derived shape primitive set.

5.4.2 Axiomatic shape primitive set

The bare minimum for an axiomatic shape primitive set is one shape primitive for each dimensionality. Restricting ourselves to Euclidean space this means four shape primitives for dimensionality 0, 1, 2 and 3:

- 
 • dimensionality 0 shape primitive: point
 The most simple shape primitive. No parameters.
- 
 • dimensionality 1 shape primitive: line
 Straight line segment, 1 parameter: *length*. All curved line segments can be obtained by embedding the straight line one-dimensional root space in another two-dimensional or three-dimensional space with a deformation component in the applied embedding mapping function. Deformation may also lead to (local) lengthening or shortening.
- 
 • dimensionality 2 shape primitive: rectangle
 Flat rectangular polygon, 2 parameters: *length* and *width*. Other flat polygon shapes can be obtained by subsequent deformations in other two-dimensional spaces and/or by aggregation (fusing) of several rectangles. Curved polygons can only be created by subsequent deformations in three-dimensional spaces.
- 
 • dimensionality 3 shape primitive: block
 Prismatic cuboid, 3 parameters: *length*, *width* and *height*. Other solid shapes can be obtained by subsequent deformations in other three-dimensional spaces and/or by aggregation (fusing) of several blocks.

5.4.3 Derived shape primitive set

There is no exact criterion to decide which shapes should be classified to the shape primitive set (axiomatic plus derived) and which not. Possible candidates:

- 1-dimensional: circular arc, circle.
- 2-dimensional: triangle, trapezium, disk (segment), shell-cone, shell-cylinder (segment), shell-sphere (segment).
- 3-dimensional: wig, frustum, solid-cone, solid-cylinder (segment), solid-sphere (segment).

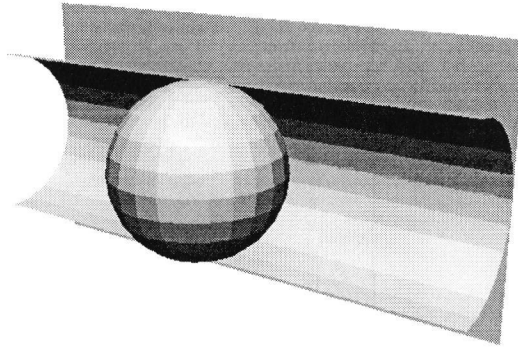


Figure 5.29 Three stages from rectangle, via semi-cylinder to sphere.

As an example: a shell-sphere. To create a sphere out of a rectangular polygon a cylindrical deformation will be applied twice. The first deformation creates a cylindrical shell with a semi-circular cross section while the second deformation creates the shell-sphere itself. The choice for the length and width values of the rectangle are directly related to the aimed radius of the sphere, that is

- $length = 2 \cdot \pi \cdot R$
- $width = \pi \cdot R$

Figure 5.30 shows the combined cell and space graph, while figure 5.29 pictures a shaded image of the three stages: rectangle, semi-cylinder and sphere. The two-dimensional root space (S1) that hosts the rectangle (F1) is cylindrically deformed (D1) in a three-dimensional space (U1) around a centre line that lies at the radius distance above the rectangle and is aligned along the length direction. The three-dimensional space U1 is similarly embedded in another three-dimensional space U2 applying the same deformation D1 along a centre

line that lies still at the radius distance above the rectangle but is now aligned along the width direction.

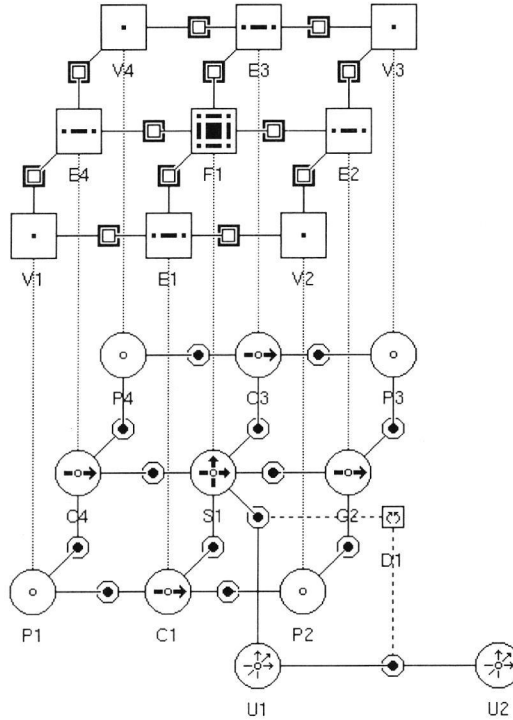


Figure 5.30 Cell and space graph of the shell-sphere primitive.

5.4.4 Shape Complex

Shape primitives are the building stones, the leaf objects of the shape graph. Other nodes in the space graph act as assembly objects and are called space complexes. The shape of a space complex is the aggregation of the shapes of the participating primitives. They can be found by traversing all the departing links of the space complex until a leaf object (= shape primitive) is encountered. Both inclusive as exclusive aggregation is suitable:

- inclusive aggregation of a set of shape primitives and/or shape complexes can be regarded as a union operation where overlapping regions are claimed more than once. Inclusive aggregation or inclusive union for two operands can be formally defined as:

$$\mathbf{P}_a \cup^i \mathbf{P}_b = (\mathbf{P}_a - \mathbf{P}_b)^a \cup (\mathbf{P}_b - \mathbf{P}_a)^b \cup (\mathbf{P}_a \cap \mathbf{P}_b)^{a,b}$$

where the suffix superscripts denote the claiming operands for that point sub-set.

- exclusive aggregation of a shape primitive or shape complex can be regarded as the combination of a difference operation and a union operation, where the intersection region is claimed only once (by the exclusive operand). This last definition put up the restriction that the exclusive operands are not allowed to overlap mutually. Exclusive aggregation or exclusive union for two operands can be formally defined as:

$$\mathbf{P}_a \cup^e \mathbf{P}_b = (\mathbf{P}_a - \mathbf{P}_b)^a \cup (\mathbf{P}_b - \mathbf{P}_a)^b \cup (\mathbf{P}_a \cap \mathbf{P}_b)^b$$

- inclusive union has precedence over exclusive union. This rule gives more modelling freedom then the alternative: a strict evaluation sequence.
- although the internal point allocation may differ for both union types, the resulting point set is the same: the union of the point sets of all participating operands.

Like a shape primitive a shape complex contains two interrelated cell and space graphs. The cell and space graphs are obtained by the union of all the cell and space graphs of the participating shape primitives and shape complexes. A new root space is added to bind all loose root spaces that resulted from this space graph union operation. This new root space is necessary to relate all par-

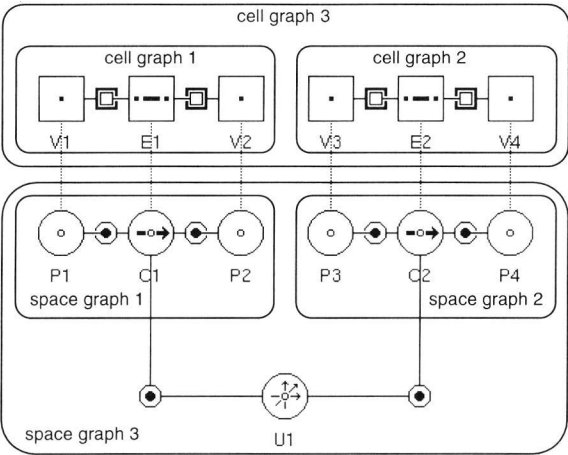


Figure 5.31 Cell and space graphs of two line primitives (1, 2) that are bundled in one shape complex (3).

ticipating parts geometrically with each other. Figure 5.31 shows the cell and space graph (3) of a space complex that bundles two line primitives (1, 2).

To demonstrate the ease the Space Deformation Tree representation at shape graph level offers to model complex shapes the modelling steps to create a ring of Moebius is selected.

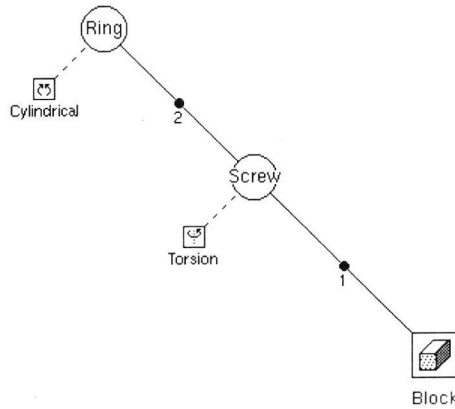


Figure 5.32 Shape graph of a ring of Moebius in three steps: block, screw, ring.

The shape graph (figure 5.32) is very simple: one shape primitive and two shape complex nodes. The dimensionality three shape primitive is a block with length $2 \cdot \pi \cdot R$, with R the required radius of the ring. The root space that hosts this block is embedded in the root space of a shape complex (Screw) and twisted by a torsion deformation. The block in this space transforms into a screw with a 180° pitch. Finally the root space of the Screw shape complex is embedded in a root space of another shape complex (Ring) and bent through a cylindrical deformation. Figure 5.33 shows the resulting shapes of each modelling step.

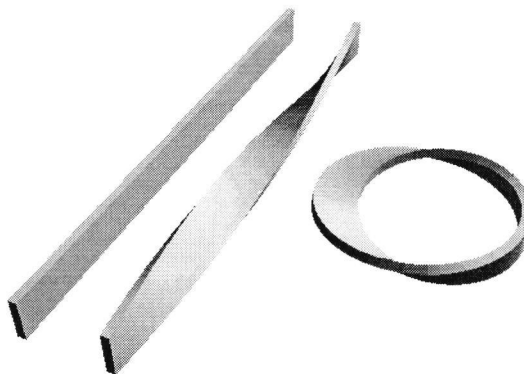


Figure 5.33 Block \rightarrow screw \rightarrow ring of Moebius.

5.4.5 Shape Occurrence

Besides the nodal objects of the shape graph (shape primitives and shape complexes) the links are also objectified: the shape occurrences. A shape occurrence places an instance of its target shape into its source shape. Thus, multiple instances of the same shape lead to as many shape occurrences. A shape occurrence is a relative occurrence of its target shape. The absolute occurrences of a certain shape can be obtained by traversing all possible paths

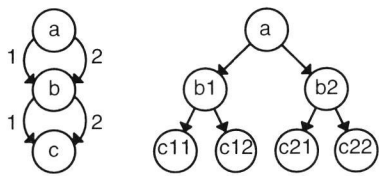


Figure 5.34 Evaluation of a shape (directed a-cyclic) graph (left) into a shape tree (right).

in the shape graph that lead to that specific shape. Figure 5.34 shows that shape *c* occurs twice in shape *b*, however, shape *b* occurs twice in shape *a* and hence four absolute occurrences of shape *c* can be found.

Obviously shape occurrences of the same shape type may differ only in loca-

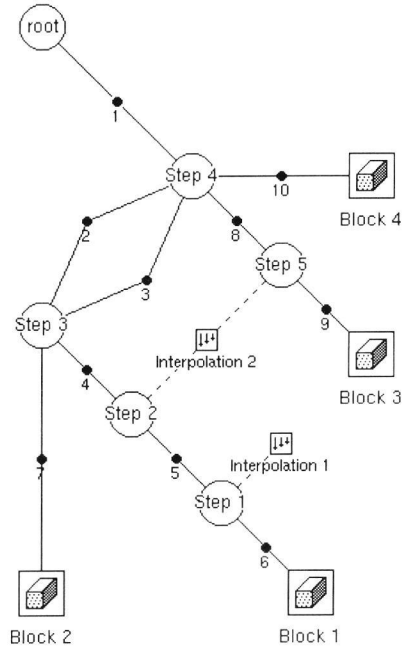


Figure 5.35 Shape graph of a bridge deck.

tion and orientation. The deformation component is attached to the shape complex itself to affect all shape occurrences that are aggregated.

To demonstrate some of the capabilities of shape graph modelling an example from the area of civil engineering is selected: a concrete bridge deck. The chosen shape can be assembled from two beam shapes (field area) connected by a block shape (support area). The beams have a flat top surface and a quadratic curved bottom surface, while the side surfaces are chamfered linearly. Figure 5.35 displays the shape graph with block 1 and block 2 as the base primitives for respectively the bottom and top part of the field area shape and similarly block 3 and block 4 for the support area shape.

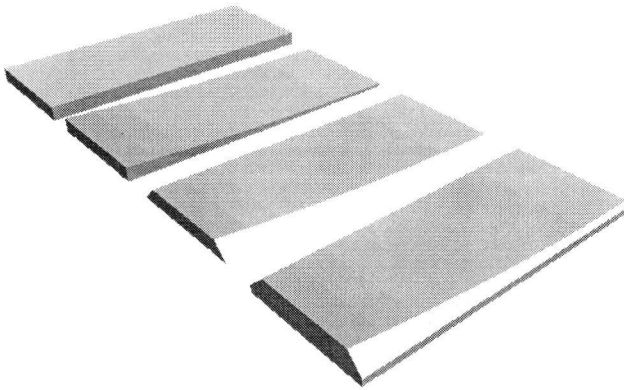


Figure 5.36 Four steps to form the field area shape of a concrete bridge deck.

Figure 5.36 displays the various steps to create the field area shape (from left to right):

- block shaped base primitive for the bottom part
- quadratic deformation on one side
- linear deformation on two sides
- aggregation with block shaped top part

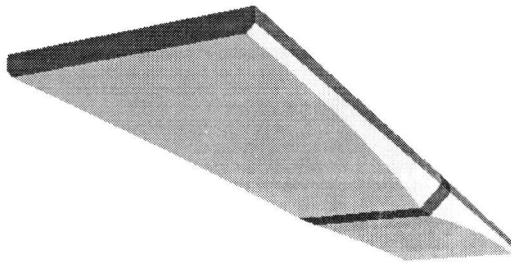


Figure 5.37 Bridge deck.

5.4.6 Shapes data model

```

graph TD
    Shape((Shape)) --> ShapeDefinition((Shape Definition))
    Shape --> ShapeOccurrence((Shape Occurrence))
    ShapeDefinition --> ShapePrimitive((Shape Primitive))
    ShapeDefinition --> ShapeComplex((Shape Complex))
    ShapeDefinition --> ShapeDeformation((Shape Deformation))
    ShapeOccurrence --> Place[Place]
    ShapeOccurrence --> Target[Target]
    ShapePrimitive --> Point((Point))
    ShapePrimitive --> Wire((Wire))
    ShapePrimitive --> Surface((Surface))
    ShapePrimitive --> Solid((Solid))
    ShapeDeformation --> Deformation[Deformation]
    ShapeDeformation --> CellGraph[Cell Graph]
    ShapeDeformation --> Space[Space]
    Point --> Circle((Circle))
    Circle --> CircleSegment((Circle Segment))
    Wire --> Line((Line))
    Surface --> Disc((Disc))
    Disc --> DiscSegment((Disc Segment))
    Surface --> ShellCylinder((Shell Cylinder))
    ShellCylinder --> ShellCylinderSegment((Shell Cylinder Segment))
    Surface --> Sphere((Sphere))
    Sphere --> SphereSegment((Sphere Segment))
    Surface --> Rectangle((Rectangle))
    Surface --> ShellSphere((Shell Sphere))
    Solid --> Block((Block))
    Solid --> Cylinder((Cylinder))
    Cylinder --> CylinderSegment((Cylinder Segment))
  
```

Figure 5.38 Data model for shapes.

5.5 Conclusions

The Space Deformation Tree representation has several characteristics that make it particular suitable for association with product models.

- the geometry part (the actual space deformation tree) offers the opportunity to define a shape object as a parametric library part in a neutral straight and flat space. Plugging the root space of such a shape into the actual space graph of a shape model will mould it according to the actual valid geometry. This technique will increase the potential domain for parametric shape objects (and therefore the possibilities of reusing previous designed objects) considerably.
- the topology part satisfies effectively non-manifold topology, which is a prerequisite for implementing explicit connectivity and reference geometry representation.
- the shape part maps nicely on various decomposition (type/occurrence) strategies.
- a space graph / cell graph combination can be considered as a sub-model (modularity) and handled as a macro-object at shape level. Integration of space graphs or cell graphs offers a flexible tool to create new shape objects out of a limited shape primitive set.
- topological embedding offers many possibilities to record spatial claims for a particular node in a product model. These claims may be disjunct or overlapping. The same facilities can be used to introduce the port concept in shape modelling.

5.6 Research Questions

At the end of chapter 4 the globally formulated research question 2:

- | |
|---|
| 2. <i>Which shape representation is able to meet the requirements of the construction industry?</i> |
|---|

was narrowed into research question 2.1:

2.1 *Which shape representation is able to meet the necessary requirements with respect to flexibility (modularity) to integrate well with the modelling structures of a product model?*

The previous section has concluded that the Space Deformation Tree representation exhibits various idiosyncrasies to fulfil the role of a flexible shape representation for product modelling.

However, research question 3:

3. *How to integrate both (product/process and shape) representations?*

determines an important additional requirement to both the product (and process) representation and the shape representation to support interoperability between the functional properties and the spatial properties.

The next chapter will discuss in detail many issues with respect to associating shape data to product (and process) data.

6

Integration of Product Modelling and Shape Modelling

There is no such thing as the shape model of a product. For example, a shape model of a building may vary between a single point (e.g. to locate it on a map) and a huge 3D-model with fine details, textures and light sources to render a highly realistic picture or virtual reality scene. Both shape representations satisfy adequately the required spatial data with respect to their application, here geographical location and realistic visualisation. A complex product can be represented by many different shapes and most of them will be generated during the design stage.

The framework shown in figure 6.1 is used to categorise all the shape models that may exist for a shorter or longer period during the design stage. It spans up three classification scales:

- *application*

First of all, what are the requirements for this particular shape or which ap-

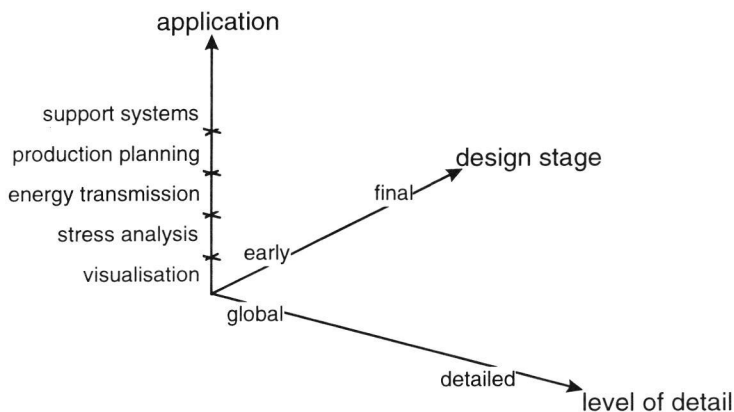


Figure 6.1 Framework for shape classification.

plication should it support? A shape model for visualisation is optimised to render a realistic picture, but is probably useless for stress calculations. Some applications need solids, others will do with surfaces and a separate thickness parameter value, and sometimes a simple bounding box is satisfactory. Occasionally applications demand for combinations of solids, surfaces and wire-frames.

- *level of detail*

Various applications need a classification to level of detail. For example, a visualisation of the total object is able to ignore the small details which dimensions are scaled below the resolution of the screen or the printer. In the VRML⁴¹ format [VRML 1996] the level of detail depends from the distance of the viewpoint to the object. If the viewpoint approaches the object a more detailed shape is loaded to compensate for the reduced distance. Often the level of detail reflects the hierarchy of a complex design. The global design at the top and detailed design at the bottom are probably done by different designers maybe working for different companies.

- *design stage*

In the early design stage designers often make use of sketches, using fat pencils to stipulate the uncertainty of their shapes. At the final design this uncertainty has shrunk until the required tolerances have been reached.

All shapes are needed, at least for a period of time, to define the product. Hence in principle, one must be able to attach each shape model to a specific hook in the product model structure. This is the central issue of this chapter:

- *how to attach many shape representations of the same product to one and the same product model, and subsequently,*
- *how to force these shape representations to conform to the product model structure and to conform to each other.*

6.1 Idealisation and accuracy

Each shape definition is an idealisation of the shape of the real world object. Even a shape model for realistic visualisation makes use of concepts like straight line, circle, sphere, flat surface, which only exist in an ideal mathemat-

⁴¹) Virtual Reality Modeling Language, a format for network exchange of virtual worlds.

ical world. The actual shape of the product that is manufactured from such an ideal *as-designed* shape model will always deviate within a certain agreed interval that is defined by the tolerance factor. This factor is traditionally a measure for the required accuracy of the manufactured product as discussed for instance in [Requicha 1983] and [Park and Lee 1996]. However, here will be demonstrated that an accuracy factor could be an important concept to integrate various shape idealisations representing the same product.

The area that should contain the actual contour of an object to satisfy the accuracy constraints can easily be constructed from the ideal (= nominal) *as-designed* shape. Two techniques can be applied:

- *spherical extent*

This method assumes a sphere with radius = accuracy around each point of the point set that is defined by the shape model. The resulting shape has an outer contour where each point keeps a minimal distance (= accuracy) of any point at the boundary of the nominal shape and an inner contour keeping the same minimal distance at the inside of the nominal shape boundary.

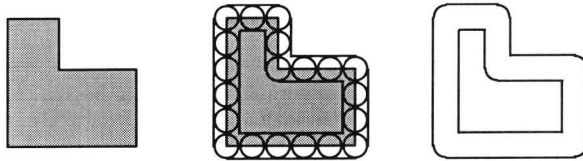


Figure 6.2 Construction of the accuracy zone (spherical extent), from left to right: the nominal shape, spheres for 'each point' of the contour of the nominal shape and the resulting accuracy zone.

- *cuboid extent*

The spherical extent approach is pure and elegant but it often leads to rather complex shapes, thus it is more practical to substitute the sphere with a cube

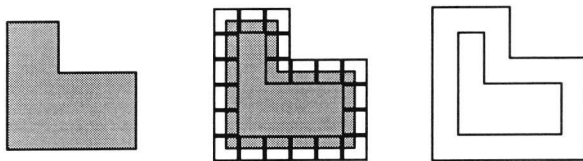


Figure 6.3 Construction of the accuracy zone (cuboid extent).

with edge-size = accuracy. The only drawback is the fact that a sphere is orientation independent while a cube needs two orientation vectors to freeze its rotational degrees of freedom. However, these orientations can almost always be derived from the orientations of the nominal shape object itself.

Now what will happen if the accuracy value⁴² is increased? The next figure shows three incremental steps in this process. The fat lined contour represents the nominal (as designed) shape. The light shaded zone represents the accuracy zone (the area that must contain the boundary of the actual shape also indicated as the *boundary zone*). The dark shaded zone represents the area that is always present in the actual shape and *never* contains the boundary (*interior zone*).

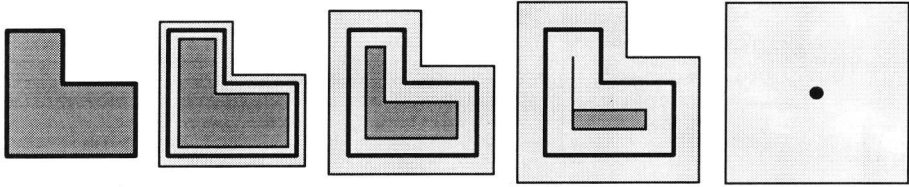


Figure 6.4 Increasing the accuracy value.

The first step illustrates the accuracy = 0 situation, all zones coincide and the boundary is completely fixed. The second and third step show that increasing the accuracy value will lead to a larger area that the actual shape may occupy but also to a smaller area that the shape definitely will occupy. The fourth step displays a situation where a part of the interior zone has been collapsed. A further increase of the accuracy value is not possible, because the overlapping boundary zones would corrupt the topology of the nominal shape. Hence, another nominal shape must be selected to allow an additional increase of the accuracy value. As the example shows, the accuracy value is upper-bounded by the smallest distance of two opposite points on the boundary of the nominal shape. In this particular example the smallest distance and the largest distance do not differ very much, which is demonstrated by the negligible remaining area of the interior zone. The last step displays that a subsequent (indefinite) increase is possible by substituting the original polygon by a single point.

Accuracy values offer several advantages:

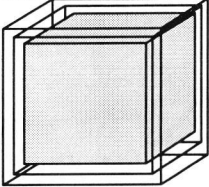
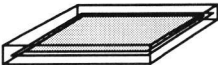


- it helps to distinguish between what is definite and what may vary and how much it may vary,
- it helps to decide which idealisations are allowed,
- it gives a criterion to compare shapes of different dimensionality,
- it is a possible representation for required shape.

The next sub-sections will elaborate the advantages mentioned above.

⁴²) To avoid confusion: increasing the accuracy value means a relaxation resulting in an expanding of the accuracy zone; decreasing the accuracy value means a contraction resulting in a collapsing of the accuracy zone.

6.1.1 Maximum and minimum shape

The table below lists the dimensionality of the maximum and minimum shapes against the dimensionality of the nominal shape.

nominal shape				
	solid	surface	wire	point
maximum	3D	3D	3D	3D
minimum	3D	2D perpendicular projection	1D radial projection	0D focus projection

Dimensionality of maximum and minimum shape.

The invariant 3D dimensionality of the maximum shape is a striking characteristic and will turn out to be an important feature for comparing nominal shapes of different dimensionality. The dimensionality of the minimal shape needs some clarification:

- Only for the *solid* nominal shape a real minimal shape can be generated.
- The minimal shape of a *surface* nominal shape is again a surface that is completely embedded by the nominal shape. However, the actual shape may claim a region that does not intersect this surface at all. In this case the interpretation of the minimal shape is that the *projection* of the actual shape on the surface completely embeds this shape.
- The minimal shape of a *wire* nominal shape is also a wire that represents the minimal interval that the actual shape (which again may claim a region without any intersection with the nominal wire) will occupy.
- Finally, there is no minimal shape of a point nominal shape.

6.1.2 Idealised shape

The accuracy value is like the resolution of a computer screen or a printer: finer details cannot be represented and should therefore be eliminated. This point is elaborated for the various nominal shape dimensionalities:

- A point cannot be further idealised as a shape, nevertheless if the distance between two points is smaller than the accuracy value they can be merged into one single point.

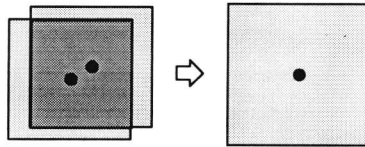


Figure 6.5 Collapsing two points, that have a distance smaller than the accuracy value, into one point.

- The length of a line (-segment) must be greater than the accuracy value. A line with a nominal length below this threshold has no minimum shape and so it does not have a guaranteed length. It can be idealised into a point shape (possibly with an increased accuracy value to compensate for the shrunken maximum shape).

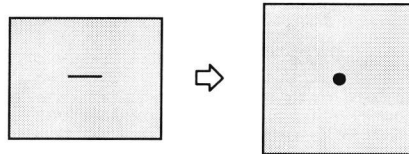


Figure 6.6. Collapsing a line, with a length smaller than the accuracy value, into a point.

- Opposite points on the boundary contour of a surface or a solid should have a distance of at least the accuracy value. If this constraint is not fulfilled the minimum shape will collapse in that area and the remaining region is not significant anymore. Idealisation by (partial) reduction of dimensionality is

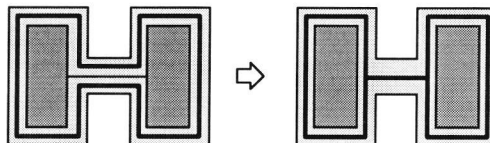


Figure 6.7 Collapsing part of a surface, that has opposite sides with a distance smaller than the accuracy value, into two surfaces and a connecting line.

justified in that situation.

6.1.3 Shapes of different dimensionality

Accuracy values facilitate criteria to compare shapes of different dimensionality. This may occur when two shape definitions claim to represent the same object or to decide if a certain shape may fit in a particular bounded space.

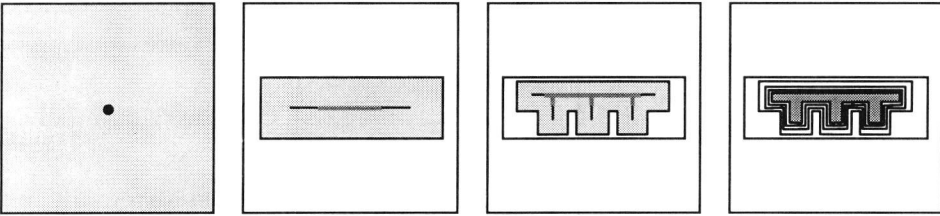


Figure 6.8 Four shape definitions for the cross section of a bridge with decreasing accuracy values.

Figure 6.8 displays four different shape definitions (presented as cross sections) of a bridge deck. The first one is a one-dimensional (line) shape. It will primarily define the minimum and maximum span of the bridge but also provides indications of the maximum width and maximum height, although the height is obviously much too large⁴³. The second shape definition is a two-dimensional (planar) shape. Compared with the first shape definition the decreased accuracy value defines the maximum height evidently more precise and also a minimum width is added. The third shape definition is an assembly of four two-dimensional planes. Now the maximum contour clearly converges to a bridge deck carried by three beams. Finally, the third shape definition determines a concrete cross sectional shape that has only a restricted variation freedom left.

Of course, a model may contain various shape definitions (referring to the same object) of the same dimensionality or constituted of shape elements of mixed dimensionalities. In this last situation the accuracy values may vary for individual elements showing that certain parts in an assembly have more or less variation freedom. This may happen if an assembly contains both to-be-designed objects and already-existing objects.

⁴³) A possible refinement is to provide different accuracy values for each orthogonal direction. It is not clear yet if the advantages of a more precise accuracy zone balance the increased complexity.

6.1.4 Required shape

In the example product and process model, described in chapter 4, shape is considered to be a sub-type entity of property. A property, in its turn, can be combined with entities of the concretisation dimension, hence: required shape, proposed shape and realised shape. From these three combinations required shape is rather unusual, how is it defined? Required shape collects a set of spatial constraints and part of these constraints can be expressed with the specification of a maximum and a minimum spatial envelope. To fulfil the requirements a (proposed) shape should define a maximum spatial envelope, which point set is a proper subset of the point set of the maximum spatial envelope of the required shape. Less formally stated, the maximum shape of the required object must entirely enclose the maximum shape of the proposed object.

If the nominal required shape and the nominal proposed shape are both three-dimensional a similar statement can be formulated with respect to the minimum shape. A (proposed) shape should define a minimum spatial envelope, which point set is a proper superset of the point set of the minimum spatial envelope of the required shape. Again less formally stated, the minimum shape of the proposed object must entirely enclose the minimum shape of the required object. If the required nominal shape has a lower dimensionality its minimum shape is interpreted as a minimum projection, i.e.:

- a two-dimensional minimum required shape demands that the perpendicular projection of the minimum proposed shape entirely encloses the minimum required shape,
- a one-dimensional minimum required shape demands that the radial projection of the minimum proposed shape entirely encloses the minimum required shape,
- a zero-dimensional minimum required shape is always fulfilled.

6.2 Shape specification

The P&P model gives a hint how to refer to shape properties: location data through an occurrence object and the shape itself through a type object. These shape property relations should be considered as an example of a possible implementation: they are formally not part of the P&P model. However, the example shape relationships are not accidentally chosen: if the application model distinguishes between occurrence and type objects this is the obvious choice. Occurrence objects will typically only differ in location (and orienta-

tion) data but it could happen that a free shape parameter still remains at this level, e.g. a weld: each occurrence will differ in length.

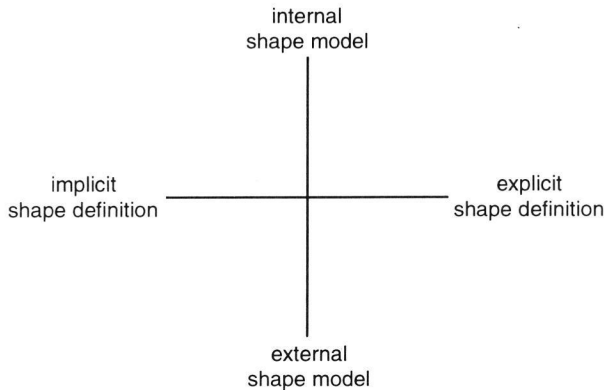


Figure 6.9 The relationship between a product model and a shape model.

The relationship between a product model and a shape model may take many forms, which can be categorised by its location on two orthogonal scales (figure 6.9):

- *internal/external shape model*
Is the shape model an integrated part of the product model or is it a model that can exist independently from the product model?
- *implicit/explicit shape definition*
Is the shape model parametrically defined, i.e. does it store only a limited number of parameter values in combination with a known evaluation routine or does it use an explicit definition, for instance a cloud of polygons?

A lot of combinations and in-between positions are feasible:

- externally defined shapes with internally specified locations,
- parametric profiles with explicit sweep curves,
- explicit 2D plan shape with implicit height data (sometimes referred as 2.5D model),
- an external explicit shape model that can be derived from an internal implicit shape model,
- an external implicit shape model, e.g. a CSG-representation,
- an internal explicit reference geometry (centre-lines and centre-planes) and attached (implicit or explicit) solid shapes (relational reference representation).

The P&P model does not exclude any combination beforehand. Nevertheless, the more interesting combinations tend to the implicitly defined internal shape models. If parametric shape modelling can be applied then this is the key to increase reusability through object-oriented inheritance techniques in combination with standard part libraries. In contrast explicit shape modelling tends to starting all over from scratch for each model instance.

The P&P model (shape) property definition can be combined with the specification dimension leading to the following combinations:

- *generic shape*

Generic shape specification is fulfilled at the meta-data level, i.e. the attribute definitions to contain the necessary parameters. A generic shape definition may, like all entity definitions, refer to other generic shape definitions and inherit part of its definition from one or more parents. Besides these parameter definitions an evaluation procedure is needed to transform the implicit shape representation into an explicit shape representation. To create such an evaluation procedure the relationship between implicit shape parameters and its explicit shape should be clear. It is hard to define this relationship formally without specifying the actual code to do the transformation, however, it is common practice to define it informally often using explanatory graphics. A well-documented test suit should complete this informal definition as a check that different evaluation procedures will produce the same result.

- *type shape*

Type shape definitions are instances of generic shape definitions. They provide or inherit parameter values in a type hierarchy until a full specification is accomplished at the leaves of the tree-like structure. Multiple (value) inheritance is allowed unless a precedence rule unambiguously decides which inherited value should prevail.

- *occurrence shape*

Occurrence shape objects share a common shape definition and normally add only location and orientation data. However, in an implementation that supports exclusively occurrence objects each occurrence shape should independently be defined.

6.3 Decomposition

The P&P model facilitates property decomposition and hence shape decomposition. It will be obvious that a (shape) property decomposition should conform to the associated product decomposition, both at conceptual level as data level. Conformance with the various decomposition structures (which are discussed in chapter 4) is elaborated in the following bullet list:

- The occurrence decomposition resembles best the internal data structure of state-of-the-art CAD systems. At the basis, groups of primitive shape objects can be assembled in a local co-ordinate system, then higher in the hierarchy shape assemblies can recursively be grouped until a single root object is encountered. To know the absolute location of a shape primitive or shape assembly all transformations between that node and the root must be coupled. The occurrence decomposition is the structure of choice if a one-to-one association with real world occurrences is imperative.
- The type decomposition is not suitable for shape decomposition because it lacks any facilities for descriptions at occurrence level. Thus, it is not able to distinguish between different positions of objects belonging to the same type.
- Mixed type/occurrence decomposition resembles best the feature based CAD systems. There is a one-to-many correspondence with real world occurrences, which are represented by each unique path between the root and a node (the structure is a single entry directed a-cyclic graph).

The previous decomposition structures are all monolithic, which means for shape decomposition that all primitive shape objects are leaf nodes while the intermediate nodes aggregate shape objects and/or other shape aggregates. Modular decomposition structures offer a natural way to introduce also shape

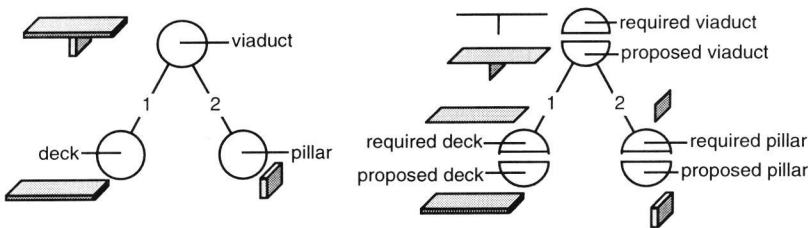


Figure 6.10 A monolithic and a modular shape tree. A monolithic shape tree declares shape objects at the leaves and shape aggregations at the other nodes, while a modular shape tree may also declare (required) shape objects and local shape aggregations at the intermediate nodes.

objects at the higher decomposition levels. The obvious way to do that is to associate shape objects with required concepts and the aggregates of shape objects with proposed concepts.

Figure 6.10 shows a small example of the shape tree of a viaduct:

- at the root: the required viaduct represented by a wire model;
- the fulfilling proposed viaduct is represented by an aggregation of surface objects;
- these surface objects represent the decomposed concepts (required) deck and pillar;
- the fulfilment of these required concepts are represented by solid objects;
- depending the applied substitution depth the viaduct can be represented as wire model, surface model, solid model or combinations of these three shape representations.

The higher level shape objects represent always the required shape. In most cases this means the maximum allowed spatial envelope the proposed shape may occupy and sometimes also the minimum spatial envelope that always should be filled.

Explicitly sharing of geometrical specifications may define additional geometric constraints. To mention a few examples:

- two linear shape objects referring to the same curve (one-dimensional modelling space) declare implicitly that both lines are co-linear,
- two planar shape objects referring to the same surface (two-dimensional modelling space) declare implicitly that both planes are co-planar,
- two straight curves referring to the same orientation vector declare they have a parallel orientation (and also the linear shape objects that refer to these curves).

Explicitly sharing of topological specifications may define additional topological constraints. To mention a few examples:

- two solid shape objects both specifying bounding faces that have an enclosure relation declare implicitly that both objects have a mating condition,
- two linear shape objects referring to the same bounding vertex declare implicitly that their ends are connected.

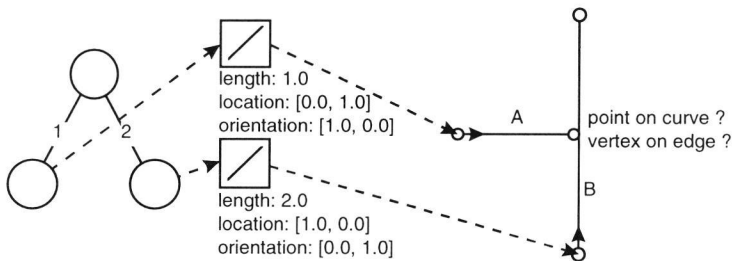


Figure 6.11 Evaluation of implicit shape definitions for additional specifications of geometric and topological constraints.

These geometric and topological constraints cannot be specified at shape level. Firstly, the implicit shape definitions must be evaluated into topological and geometric representations. Secondly, these separate representations must be integrated into one representation: an inter-linked topological graph and geometric graph. Integration software will find all kinds of potential relationships and representation items that may be shared. The designer should distinguish between the relationships and sharable items that are intentional and those that are coincidental.

Topological constraints and connectivity are subjects that are closely related. The next section will focus on the association between product structure connectivity and connectivity between shape objects.

6.4 Connectivity: Port shape

Decomposition of a complex shape object into a set of more atomic shape objects can be handled without paying attention to connectivity issues. Each shape object is located and oriented individually, fully independent of all the other shape objects. If it is done accurately, visual inspection will not give a clue if connectivity data is available. Nevertheless, there is a deficiency in structural data that will manifest itself if there is a need to know if and how part A is connected to part B⁴⁴. Usually, there is connectivity data in the product structure, but these specifications are global and will definitely need additional specifications in the shape structure.

⁴⁴) This information cannot be derived from the geometric data. Because of the computer representation of the real number set (which is not continuous but a finite set of discrete numbers) equality tests, especially after computation, are useless. Besides this mathematical objection there is also a semantic objection: from the fact that objects are located near to each other cannot be deduced that they are physically connected.

In the P&P model (modular) connectivity between two concept objects is achieved by two mating port objects (one for each concept object). A port object is a potential connection point, which means that the shape equivalent of a port should be something of 'a potential connection area'. A connection assumes a physical contact between two shape objects and a physical contact can be described with a topological relation. In a clean set theoretic approach connecting two shapes can be interpreted as the union of their point sets.

$$\mathbf{P} (a \text{ connected to } b) = \mathbf{P} (a) \cup \mathbf{P} (b)$$

However, the inverse operation (disconnecting) must be realised by dividing the elements of the combined point sets, i.e. without duplicating point elements, hence

$$\mathbf{P} (a) \cap \mathbf{P} (b) = \emptyset$$

This means that there are no point elements of two connected shapes that belong to both shapes. This can be realised by excluding the connection boundary of one shape operand as described in chapter 5. Another possibility is to exclude both connection sides and add a third 'connection' shape that fills the intersection points of the two shape operands. The semantics of a two-operand connection is that one operand is not affected by the connection (at least at that level of description) and that the other operand must conform to the first operand. The semantics of the three-operand connection is a more balanced situation where both shape operands conform to each other. Another application is concurrent engineering: inclusion and exclusion could dictate a precedence of the interface design, i.e. the included side is designed first the excluded side is designed afterwards in conformance with the included side. The interpretation of the three-operand connection: the interface itself is designed first then later both sides are designed in conformance with the predefined interface.

Of course, this set theoretic approach of connected shapes is only relevant in combination with the already discussed accuracy factor and the use of reduced dimensionality shapes at the more global levels of the product/shape tree. It is here that the material division of the shape operands over the connection is not yet decided, however, at the leaves (applying a very small accuracy value) this material division should be unambiguously defined.

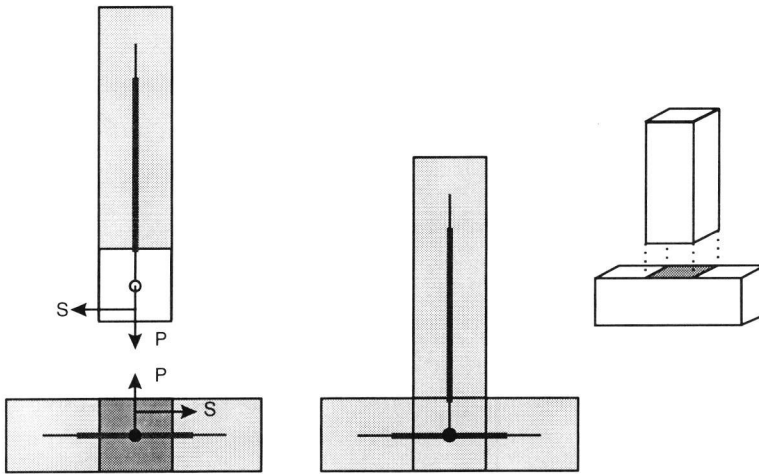


Figure 6.12 Connectivity (disconnected and connected) between two linear shaped objects. The bottom part has an inclusive port, while the top part has an exclusive port. The right picture shows a possible solution at a more detailed level.

For example, if two edges referring to different curves are connected (figure 6.12) then the procedure is as follows:

- determine the connection point, i.e. the point that would be the result of an intersection operation on both point sets,
- mark this point in both point sets and attach a vertex to each,
- determine which point set should exclude this point,
- let the port objects reference these vertices,
- establish the connection⁴⁵ both at product model level and shape model level.

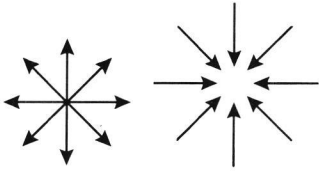
This procedure assumes that both shape operands are located and orientated independently from each other. Another approach could be to start with one or more fixed ‘anchor’ objects and assemble the other parts by connecting their ports. In that case a port needs additional orientation specifications by specifying one or two orientation vectors. A primary orientation vector points to the expected neighbouring point(s) of ‘the other side’. A secondary orientation vector freezes the rotational degrees of freedom that may remain after the em-

⁴⁵) Port objects and their spatial representations can be defined without specifying the actual connection. This is an essential requirement for modularity.

ployment of the primary orientation⁴⁶. Obviously, the components of a primary orientation vector need not be fully specified, e.g.:

$\mathbf{p} = [\pm 1, 0, 0]$ 

means orientation parallel to the x-axis (and a rotational degree of freedom around this primary axis), or

$\mathbf{p} = [\sin \varphi, \cos \varphi, 0]$ with $-\pi/2 \leq \varphi \leq \pi/2$ 

means the orientation is parallel to the x/y plane (and a rotational degree of freedom around the z-axis). Theoretically even omni-directional orientation is feasible.

The primary orientations of mating ports are diametrical, which lead in the examples above to two variants: an inward and an outward orientation.

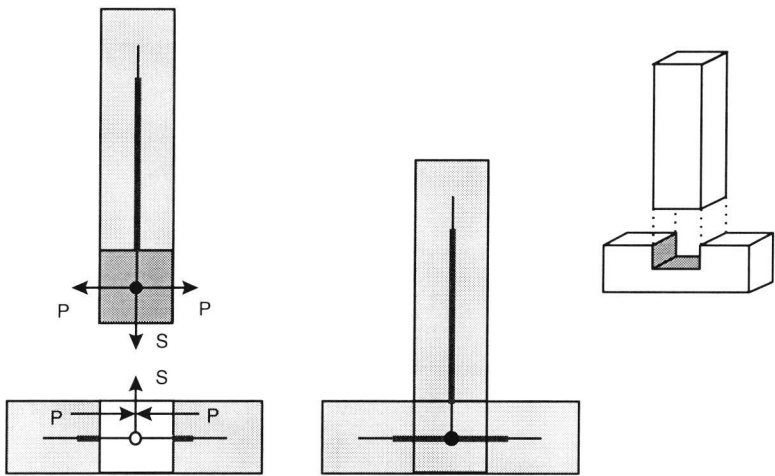


Figure 6.13 Connectivity (disconnected and connected) between two linear shaped objects. The bottom part has an exclusive port, while the top part has an inclusive port. The right picture shows a possible solution at a more detailed level.

Figure 6.13 illustrates another solution for the two connecting linear objects. Here the connection point is included by the top part and excluded by the bot-

⁴⁶) The secondary orientation vector may be omitted in case of a rotation invariant shape operand (e.g. a cylinder).

tom part. Because the port of the bottom part is ‘somewhere in the middle’ it is surrounded by two neighbouring points and hence the double primary orientation.

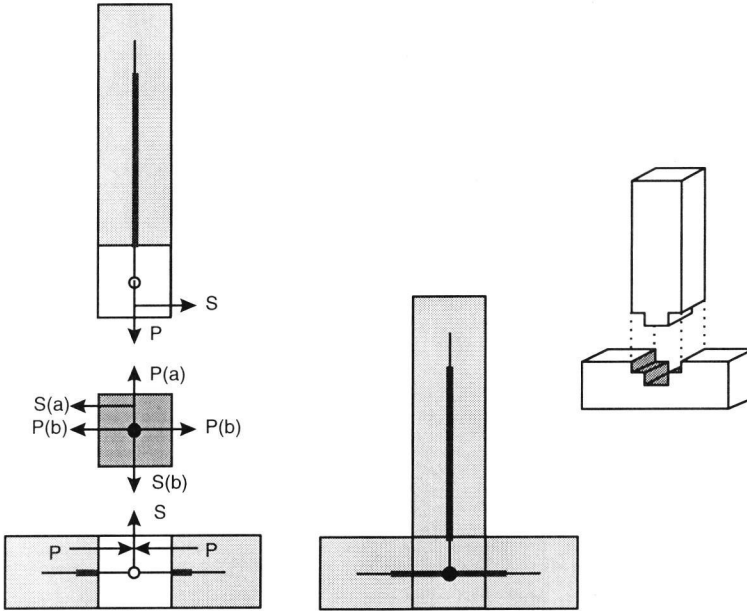


Figure 6.14 Connectivity (disconnected and connected) between two linear shaped objects. Both top and bottom parts have exclusive ports, while an additional middle part has the inclusive port. The right picture shows a possible solution at a more detailed level.

Figure 6.14 illustrates yet another solution for the two connecting linear objects. Here the connection point is included by a separate middle part and is excluded by the bottom and top parts. This is the most general solution especially if the connection is a product in itself, i.e. using additional components that are not available in either of the two operands.

Up to this point the examples all show linear shaped operands and point shaped ports. This is done for explanation only for there are no restrictions to the dimensionalities of the operands. The point set that defines the shape of a port object must be a subset of the point set that defines the shape of the concept object that owns the port.

$$\mathbf{P}_{\text{port}} \subseteq \mathbf{P}_{\text{concept}}$$

Implicitly, this also restricts the dimensionality of the port object to lesser or equal the dimensionality of the concept object.

$$\text{Dim (port)} \leq \text{Dim (concept)}$$

As a result, the earlier suggested recipe to determine port shapes could be generalised as follows:

- determine the point set that would be the result of an intersection operation on the point sets of the concept operands.

$$\mathbf{P}_{\text{intersection}} = \mathbf{P}(a) \cap \mathbf{P}(b)$$

- let one or more⁴⁷ appropriate topological entities refer to the intersection point sub-set in each operand point set.

$$\mathbf{P}_{\text{intersection}} = \mathbf{P}_{\text{top},1} \cup \dots \mathbf{P}_{\text{top},i} \cup \mathbf{P}_{\text{top},i+1} \cup \dots \cup \mathbf{P}_{\text{top},n}$$

- determine which side should include and which side should exclude (or both sides exclude under insertion of a third connection operand).
- let the port objects reference the defined topological entities.

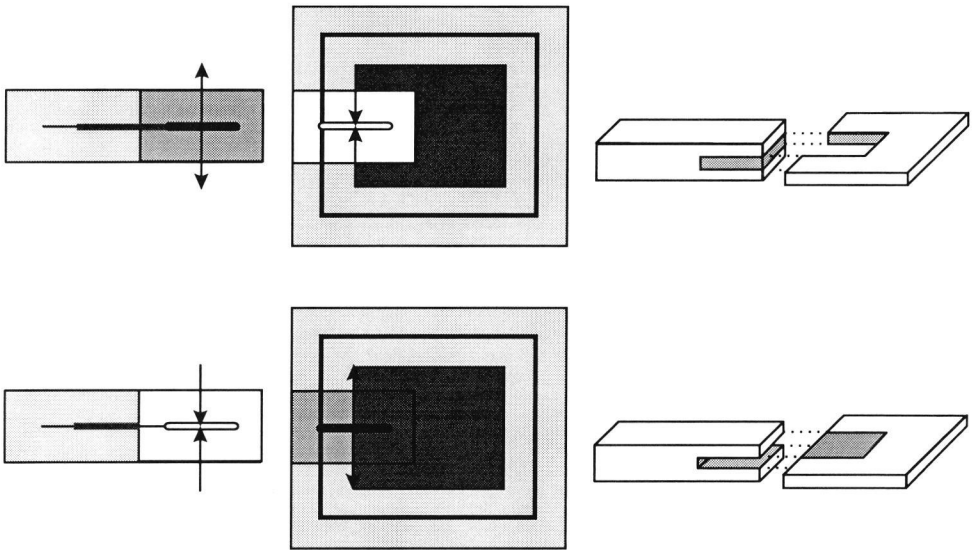


Figure 6.15 Connectivity example between a linear shaped object and a planar shaped object.

⁴⁷ Because the intersection point sub-set need not be contiguous (or continuous), more than one topological (root) entity may be necessary.

6.5 Port decomposition

The P&P model facilitates both concept decomposition and port decomposition, although there are certain constraints on port decomposition to guarantee the dependency relation between a port object and the concept object that owns the port:

- The sub-ports that constitute the decomposition of a port A should all belong to one or more of the sub-concepts that constitute the decomposition of the concept B that owns port A .
- These sub-ports are all so-called *free ports*. They are not (and cannot be) mated with a port of another sub-concept of the same decomposition cluster.

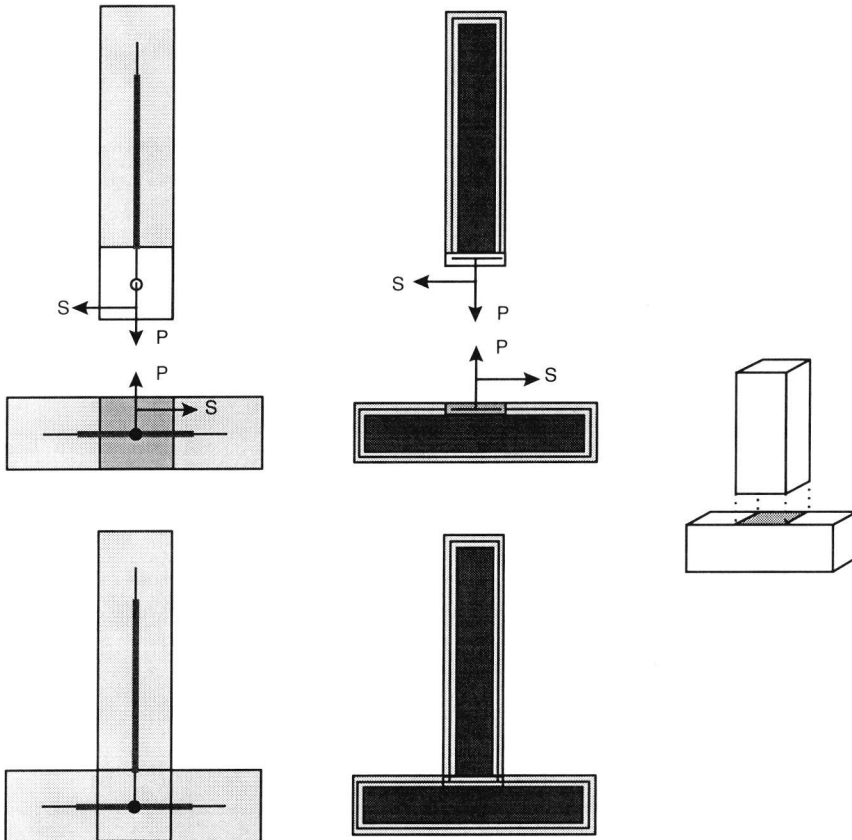


Figure 6.16 Two different levels of detail to represent the same concept objects and port objects before and after the connection. The left figures show two linear shaped concept objects and point shaped port objects, the right figures show two planar (or solid) shaped concept objects and linear (or planar) shaped port objects.

The idea is that connections between the sub-concept objects constitute the internal structure of the super-concept object. However a port of the super-concept object attaches with the external environment and hence the constituting sub-ports should do the same.

With regard to the shape of a port an extra constraint must be introduced:

- The merged point sets of the maximum shapes of the sub-ports must be a sub-set of the maximum shape point set of the super-port. The rationale is that shapes on a lower level of decomposition expose more detail and, hence, use a smaller accuracy value than shapes higher in the decomposition tree.

6.6 Conclusions

A complex product can be represented by many different shape models, though only one product model. The product model will normally show a hierarchical structure: global information near the root and detailed information near the leaves. This hierarchical product information structure is also convenient to locate the different shape models, distributing their sets of shape elements over the leaves (and sometimes also the intermediate nodes) of the product tree. Traditionally, hardly any constraints are imposed to these shape models, which reduce their interrelations to the single fact that they are attached to the same product model. This situation with numerous independent shape models is unavoidable in an information system with so-called islands of information, however it is unacceptable for an integrated information system.

Criteria must be stated to help to compare two shape models and to decide if they are consistent with each other and their common product model structure:

- *accuracy*
in the sense as introduced and defined in this chapter. By stating that the point set of the maximum shape of the model with the larger accuracy value must be the super set of the point set of the maximum shape of the model with the smaller accuracy value, a natural kind of a global / detail hierarchy has been introduced⁴⁸.
- *harmonised decomposition*
a shape model can also be structured along a decomposition tree. By coupling the shape decomposition to the product structure decomposition the

⁴⁸ An important advantage of maximum shape is the fact that it is always represented by solid elements, which makes it very convenient to compare shape models with elements of mixed dimensionalities.

shape model offers as many referencable entries as there are nodes in the tree, instead of only one entry: the root node. If all shape models are aligned following the same decomposition an important requirement for comparison is fulfilled.

- *explicit connectivity*
explicit connectivity is another criterion to compare the consistency of shape models.

Many of the concepts and solutions that are discussed in this chapter are still ‘under development’. The projects, described in the next two implementation chapters, were not aimed at the issues of integration of product modelling and shape modelling. As a result, the case projects could not offer the opportunity to test the implementation of these integration constructs beyond the level of academic proto-typing.

However, the constructs described in chapter 4 (Product modelling structures) and especially the constructs described in chapter 5 (Shape modelling) could be tested thoroughly. The next two chapters will focus on the development of information models for highway design and viaduct design.

7

Case 1: Roads

In product data technology, road design has always played an important role. Yet, the development of an information model for roads that supports a higher level of semantics than a bare minimum of explicit geometry (e.g. a labelled set of 3D strings) is not a trivial task. This chapter discusses the features and issues of such an information model: the Road Shape Model Kernel (RSMK). The chapter also describes how the RSMK can be evaluated into a Space Deformation Tree (SDT) representation.

7.1 Introduction

7.1.1 Historical overview

Around 1990 Rijkswaterstaat and TNO initiated the development of a product model for highways that resulted in several versions of the so-called Road Model Kernel (in Dutch: BasisWegModel) [Willems 1990a]. This first attempt had an ambitious scope: ultimately it should deal with all the product life cycle stages and it should address a broad scope from the road network at the top level until the road furniture at the bottom level. The road model was probably the first product type model that extensively utilised a number of information modelling principles of the General AEC Reference Model (GARM) [Gielingh 1988]. For example, it incorporated the explicit distinction between function (requirements) and the chosen technical solution to fulfil this function (characteristics). Besides, STEP methods and tools were applied, e.g. the use of the EXPRESS information definition language and the STEP physical file format.

Over the years several attempts have been undertaken to propose this project for international standardisation as:

- an application protocol within the ISO/STEP community,
- a work item in the PIARC organisation and
- a proposal in a European technology project (Brite EuRam).

So far these efforts were all unsuccessful. Still the issue is brought up regularly in the STEP/AEC sub-committee, lately by the Swedish and German delegations.

The implementation of translators to connect road design applications (of the 1970's and 1980's) to this exchange format for road data were obstructed by the large gaps in scopes, semantics and shape representations. The applied road design applications were:

- *WEGEN*

WEGEN was⁴⁹ an in-house developed package of many autonomously operating programs that communicated through various ASCII files with each other. This absence of a central project database was the major obstacle for a reliable interface. On the other hand did the separate modules for horizontal and vertical alignment and the road cross section map nicely to corresponding model constructs in the Road Model Kernel.

- *MOSS*

MOSS (Modelling SystemS)⁵⁰ is a commercial package that originally was developed as surface modeller for digital terrain modelling (DTM). This history was preserved in the conceptual model of the internal database and the offered input and output facilities. The road design data was stored in a set of so-called 3D strings: lists of 3D points with associated attributes (e.g. gradients) and an identification label. This representation appeared too explicit to be considered for translation to the Road Model Kernel.

As a result, only a small portion of the model could be filled with the data the road design applications could provide. At last the development of the Road Model Kernel was terminated in 1994.

In this same year a new project was started to develop a road model with a much more limited scope. It should address road design data especially shape data, however without intersections, in other words a simple road as its top entity. The functionality should be sufficient to fulfil the input requirements of CAD-systems that are applied to design special civil engineering structures like fly-overs and tunnels. The remainder of this chapter will describe this road model (Road Shape Model Kernel) [Willems 1995] and the procedure to derive an explicit shape representation from its compact hierarchical data structure.

⁴⁹ The development is in the mean time terminated.

⁵⁰ Mark this section discusses a version of MOSS as available in the early nineties. The current version of MOSS will probably differ on many points from this version.

7.1.2 Modelling issues

From a modelling perspective, roads, like other infra-structural systems (e.g. railways, dikes), form a special category of civil engineering products. They:

- a. have a very special *shape* with a single dominant dimension (longitudinal direction),
- b. show a less obvious *compositional* breakdown (i.e. more monolithic than assembly/parts structure),
- c. lack a distinct unambiguous contour that separates the object of interest from its *environment* (no clear-cut interfaces).

- *Shape*

The design process of a road consists of two sub-processes. One sub-process deals with the design of the alignment of the road resulting in a horizontal alignment (the projection of a guiding curve on the horizontal plane). It also handles the one or more vertical alignments (the projection of the height curve on the vertical surface intersected by the horizontal alignment). The other sub-process deals with the design of the road itself by its transversal definition: the different parts of its cross-section and its longitudinal definition: variations of the cross-section along the road.

This means that the alignment sub-process defines the geometry to be used as representation by the cross-section sub-process. This mixing of definitions and representations tend to lead to rather messy models.

- *Composition*

Roads cannot, in contrast to steel structures, be assembled from distinct parts linked together with sharp defined connections. They show a more integrated monolithic structure. Still the average road design is too complex to do without any form of decomposition. This complexity is two-fold:

- firstly, it is caused by the impact of the horizontal alignment and vertical alignments on the total road design (until then geometrically considered as straight and flat) and
- secondly, the accumulated variations in longitudinal and transversal directions.

An extra complication forms another kind of ‘decomposition’ by which alignments are assembled from element parts like straight elements or circular arc elements. In principle, the alignment decompositions are independent and therefore not in line with the road-as-a-product decomposition. Attempts to harmonise them nevertheless must lead to reasonably inefficient

model structures.

- *Environment*

The road and its environment are tightly interlinked: a road cannot be designed without detailed knowledge of its aimed environment, while the environment is essentially transformed after the realisation of the road design.

This interlinking between the road model and its environment (digital terrain model) imposes a major modelling problem. Of course, the terrain model could be incorporated into the road model but they are very different in nature ('designed' data versus 'natural' data or implicit parametric representations versus explicit grid or string representations). A real interaction is only feasible between an evaluated and more explicit representation of the road shape and the terrain model representation. Besides, more applications than road design do make use of terrain data therefore it seems more appropriate to create only one terrain model that can be shared and updated by different applications.

7.2 Road Shape Model Kernel

The principal requirements for the Road Shape Model Kernel (RSMK) were aimed at the data exchange between road design applications and CAD-systems for the design of special civil engineering objects (fly-overs, tunnels) in a particular road design. The point was to find an interface in the road design process with a manifest demand for product data exchange while at the same time a limited scope was needed to keep the project manageable. Both requirements were fulfilled in this case:

- Especially the low level representations that used to be exchanged hamper the fly-over/tunnel designer because he is not able to make even minor changes in for instance a vertical alignment.
- The limited scope permits disregarding the link with the terrain model, intersections⁵¹ and road furniture.

⁵¹) Strictly intersections may happen (and sometimes will happen) within a fly-over or tunnel structure. This is still rather exceptional and therefore ignored for the scope of the RSMK.

This chapter discusses the modelling structures and some of the design decisions that were taken to develop the RSMK. A detailed documentation of the model presented as an annotated EXPRESS model (included as appendix) in either text or HTML format can be obtained from Rijkswaterstaat or TNO.

The RSMK is defined in five sets of closely related entity types: the so-called Units of Functionality (UoF):

- *Chain UoF*
A generic domain for all sorts of chain-like data structures.
 - *Horizontal and Vertical Alignment UoF*
An information domain that describes the projection of the central guiding curve on a horizontal plane and the projection of a height curve on a vertical surface that is defined by the horizontal alignment and the gravitational axis.
 - *Road UoF*
An information domain that describes the road without the influence of any alignment, in other words straight and flat.
 - *Super Elevation Alignment UoF*
An information domain that describes specific transversal gradient deviations.
 - *Width Alignment UoF*
An information domain that describes specific transversal size deviations.
- These units of functionality are discussed in the following subsections.

7.2.1 Chains

Chain structures appear to be very dominant in road modelling: the horizontal and vertical alignment show a chain structure of succeeding geometry

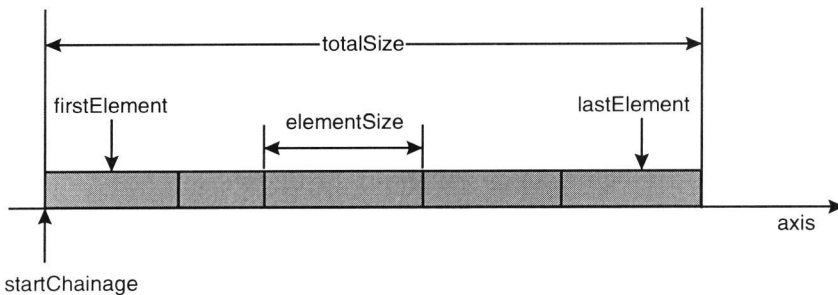


Figure 7.1 Chain assembled from several contiguous elements.

elements, a cross-section consists of adjacent linked elements and subsequent cross-sections can also be stringed like beads to a chain. The general linked list structure is not sufficient but can be used as a base class to form the abstract super-type chain⁵².

The chain domain contains two top-entities: *chain* and *chainelement*. One or more linked chainelements may compose a chain, if

- exactly one element has no predecessor (first element)
- exactly one element has no successor (last element)
- each element is visited only once traversing from the first element to the last element.

These rules guarantee a proper chain without loops or gaps. A chain is always related to a one-dimensional axis oriented in longitudinal or transversal direction. The chain positions the beginning of its first element at a fixed point (*startchainage*) while the intermediate and last ordinates can be calculated with the sizes of the subsequent elements.

7.2.2 Horizontal and vertical alignment

There is a dependency between a horizontal alignment and one or more vertical alignments. A vertical alignment only makes sense in combination with a horizontal alignment because it defines a height curve while tracing the curve of the horizontal alignment. For this reason the vertical alignment definition is only valid for that part that resides within the definition interval of the matching horizontal alignment. The particularities of coupling a vertical alignment to a horizontal alignment are discussed in section 2.3.

Accuracy

The best way to define a horizontal (or vertical) alignment has always led to much debate. The first models aimed at a definition that was completely free of any redundancy. The idea was to establish one anchor location and use that location as a starting point to calculate, by integrating an explicit stored curvature function, progressively the successive locations. An inherent strong point of this approach is the prevention of any inconsistencies between the stored values: no value can be calculated applying one or more other values. However, a weak point is the accumulation of errors for each subsequent step in the calculation. In another model version it was attempted to solve this disadvantage by introducing a system of deliberately applied redundancy that was

⁵²) An abstract supertype is a type that is not supposed to be instanced directly but only through its descendants.

intended to control the quality of the curvature integration calculation. Yet, the resulting model was rather complex. Besides, if the data items are stored with sufficient accuracy the problem of error accumulation is not really an issue. Finally, the original approach was selected.

Horizontal Alignment

The horizontal alignment is a compound of a set of curvature⁵³ functions defined on adjacent non-overlapping intervals. The RSMK distinguishes three kinds of polynomial curvature functions of type $f : ax + b$

- $a = 0 \wedge b = 0$
A constant curvature of zero, i.e. a straight section.
- $a = 0 \wedge b \neq 0$
A constant non-zero curvature, i.e. a circular section.
- $a \neq 0$
A linear curvature, i.e. a transition section (clothoid).

This set of curvature functions must be considered as a minimum set. The question which curvature functions should be included is a basic information modelling issue that is certainly not typical for road modelling. It all has to do

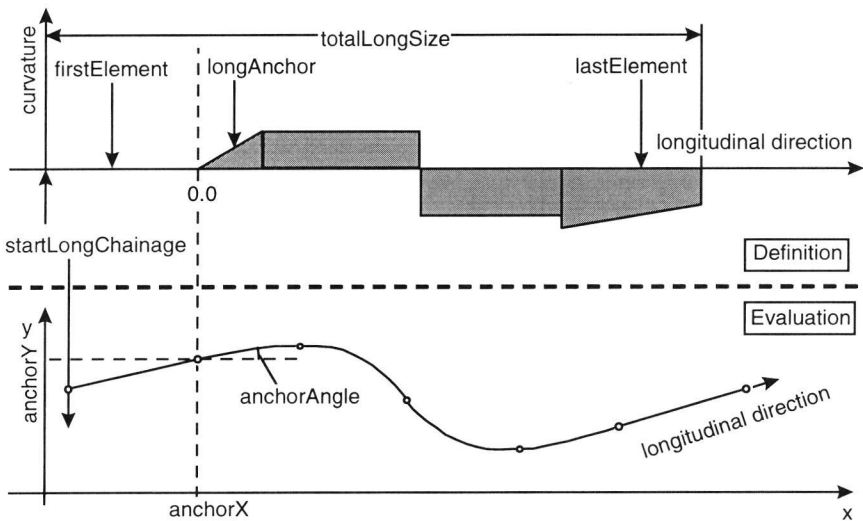


Figure 7.2 Definition and evaluation of a horizontal alignment.

⁵³) Curvatures are preferred over radii ($R = 1 / \rho$) because radii have an infinite value in very common circumstances, e.g. in a straight section.

with the accuracy that can be achieved with conversions from one set of curvature functions to another. A large set of curvature functions is convenient for applications that use the same function set, but increases the necessary conversions to perform by the post-processor for applications that use dissimilar function sets. However, a small set localises these conversions at the pre-processor side. This seems more appropriate because in that situation the model (standard) dictates which conversions must be implemented while otherwise each new application could change the situation.

To evaluate the horizontal alignment definition into an explicit curve the next procedure must be executed:

1. Start with the anchor point⁵⁴ co-ordinates x_0 and y_0 and the anchor angle φ_0 , stored at horizontal alignment level,
2. Integrate the curvature function to calculate the Δx , Δy and $\Delta\varphi$.
3. Compute the new location and orientation⁵⁵: $x_1 = x_0 + \Delta x$, ...
4. More elements? Repeat step 2 and step 3 ...

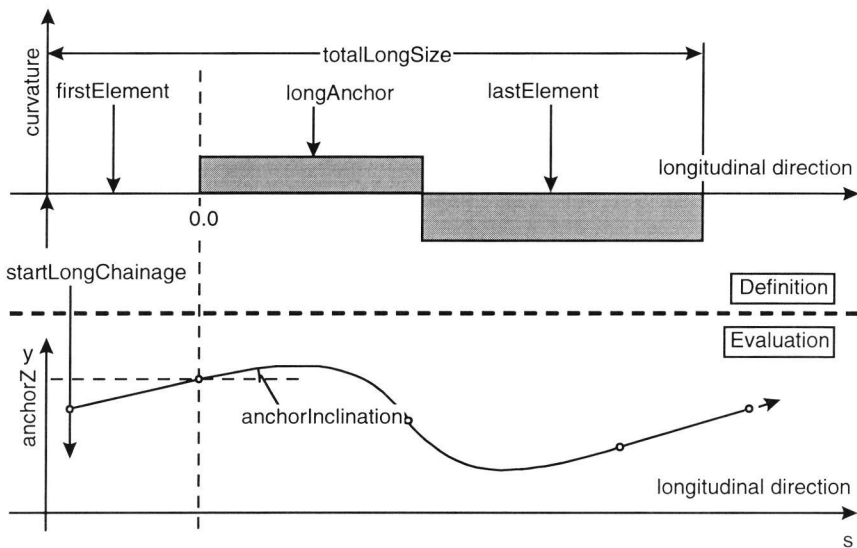


Figure 7.3 Definition and evaluation of a vertical alignment.

⁵⁴) The alignment anchor need not necessarily refer to the start of the alignment interval.

⁵⁵) This can be done because the RSMK has hard-wired two connection conditions between two elements: 1. continuity i.e. no gaps and 2. smoothness i.e. the tangents at both sides are the same. However, the curvatures may have different values at both sides (curvature leap).

Vertical Alignment

Much of what is said concerning the horizontal alignment also holds for the vertical alignment. Nevertheless, there are a few differences:

- Because of the small curvature values (large radii) that are common for the vertical alignment there is no need for a transition element.
- Because the vertical alignment traces the curve of the horizontal alignment the evaluation of the explicit curve shows some deviations. The actual length of the evaluated curve will be longer than the ‘official’ length of the alignment. Yet, this peculiarity is common practice for road design.
- The anchor of a vertical alignment consists of a height co-ordinate z_0 and an inclination $\tan \theta_0$.

7.2.3 Road

The structure of the Road Unit of Functionality forms the backbone of the RSMK. It defines the decomposition structure and the hooks that may be connected to the different alignments (alignment in a broader sense, i.e. horizontal and vertical alignment extended with ‘super elevation alignment’ and ‘width alignment’). The level of the alignment references determines the influence scope of this particular alignment, i.e. near the root: global significance, near a leaf: local significance.

The decomposition strategy of the RSMK coincides with the principal directions, i.e. first a longitudinal decomposition and then two transversal decompositions. The root entity is *road*, it is the single entry from which all other data objects can be reached. At this global level the horizontal alignment is plugged in, which implies that the influence of the horizontal alignment affects all elements of the road structure.

Longitudinal decomposition

The longitudinal decomposition divides the road in a contiguous set of *road sections* (borrowing the chain structure through inheritance: road from chain and road section from chain element).

For each road section is recorded its:

- longitudinal size, i.e. the length measured along the horizontal alignment curve,

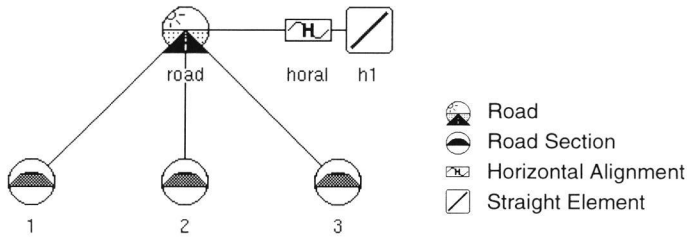


Figure 7.4.a Longitudinal decomposition of a road into road sections.

- transversal position with regard to the horizontal alignment curve. In this manner a road section may transversally determine its position. This choice ensures maximum flexibility but the prize is that the model cannot guarantee the continuity of for instance a carriage way at both sides of a road section boundary⁵⁶. Yet, hard-wiring this continuity in the model, if already possible, tend to lead to complex structures while the advantages are dubious. For instance, which continuity is intended in the figure 7.4b?

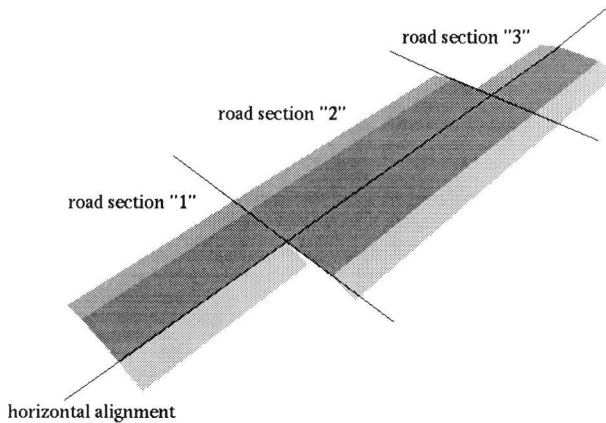


Figure 7.4.b Evaluation of example of figure 7.4.a.

Transversal decomposition

The first transversal decomposition divides a road section into a contiguous set of *road elements*. A road element is an abstract super-type of two entities: *road element complex* and *road element primitive*. The road element complex is further decomposed (the second transversal decomposition) into a contiguous set of (again) road element primitives. A road element primitive is an abstract

⁵⁶) Of course, an EXPRESS rule may help here.

super-type of a road functional unit: carriage way, verge and slope. The reason for introducing the extra aggregation entity road element complex is entirely to determine a group of road element primitives that share the same vertical alignment. This implies that a road section may contain several parts, each independently referring to a vertical alignment. Obviously the vertical alignments are referenced from the road element complex level.

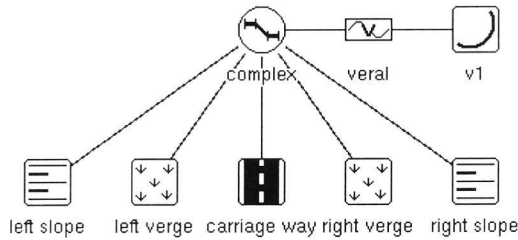


Figure 7.5.a Transversal decomposition of a road element complex into a set of road element primitives.

For each road element complex is recorded its:

- transversal position of the vertical alignment curve. In this manner a road section complex may determine the position where the vertical alignment lifts this substructure.

For each road element primitive is recorded its:

- transversal size: the dimension measured perpendicular on to the horizontal alignment curve,
- transversal gradient: the tangent of the plane in transversal direction.

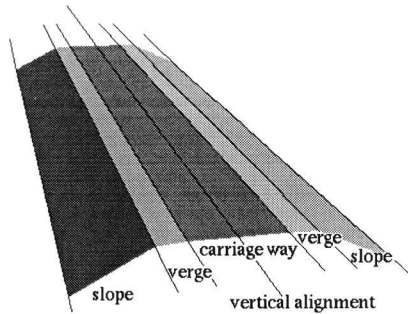


Figure 7.5.b Evaluation of example of figure 7.5.a.

If a road section contains two road element complexes (with probably two different vertical alignments) a transition road element is necessary to bridge the gap between the height variations. The transition can be achieved in two ways:

- define the width (possibly with variations) and calculate the transversal gradient, this is the usual choice for a central reserve verge, or
- define the transversal gradient (> 0) and calculate the width, which is common practice for slopes. The gradient value is interpreted as an absolute value because sign of the height differences may vary.

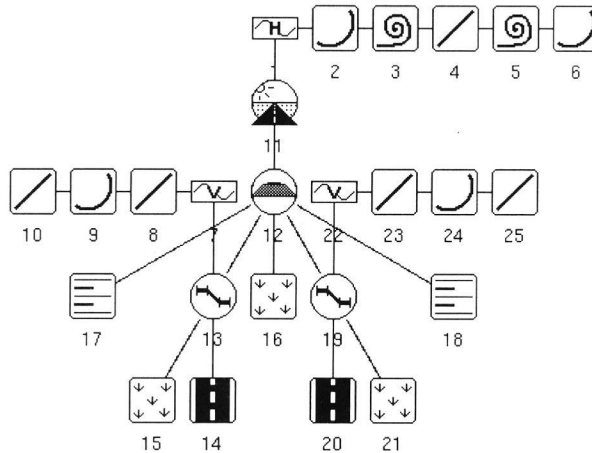


Figure 7.6.a Road with two road element complexes (13, 19), which refer to separate vertical alignments (7, 22), and a central reserve verge (16) acting as transition element to bridge the height differences.

The figure above shows a simple but complete example of a RSMK model instance. Notice the complete separation of the product structure (road, road section, road element complex, road element primitive) and the various alignments (road geometry). This indicates that the RSMK road structure is suitable to implement standard part libraries. The geometry can be plugged in later, at the moment the library object is actually used.

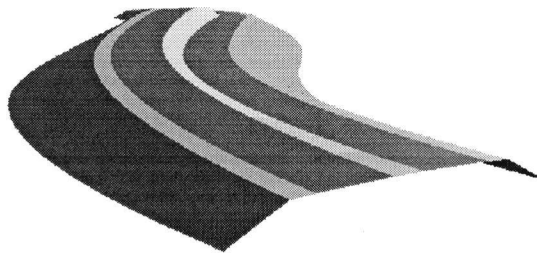


Figure 7.6.b Evaluation of example of figure 7.6.a.

7.2.4 Super elevation and width variations

The previous section mentioned the transversal size and transversal gradient of a road element primitive (carriage way, verge or slope) as a fixed scalar value. These attributes are allowed to vary with the longitudinal location. A variation of the transversal gradient is called the *super elevation alignment*, while a variation of the transversal size is called a *width alignment*. Both alignments affect only the referring road element primitive however adjacent road element primitives may be influenced by shifting in transversal and/or gravitational direction. The super elevation and width alignments show a similar structure as the horizontal and vertical alignments: along the longitudinal axis a set of elements determines contiguous intervals of constant or changing elevation/width values.

Super elevation alignment

The super elevation alignment controls the variation of the transversal gradient along a certain interval on the longitudinal axis. In normal practice the super elevation alignment is only applied to carriage ways. Other road element primitives may show changes in their transversal gradients when they act as transition elements with a predefined width. However, this is derived from the height conditions of the neighbouring road element complexes and is not a priori defined using a super elevation alignment.

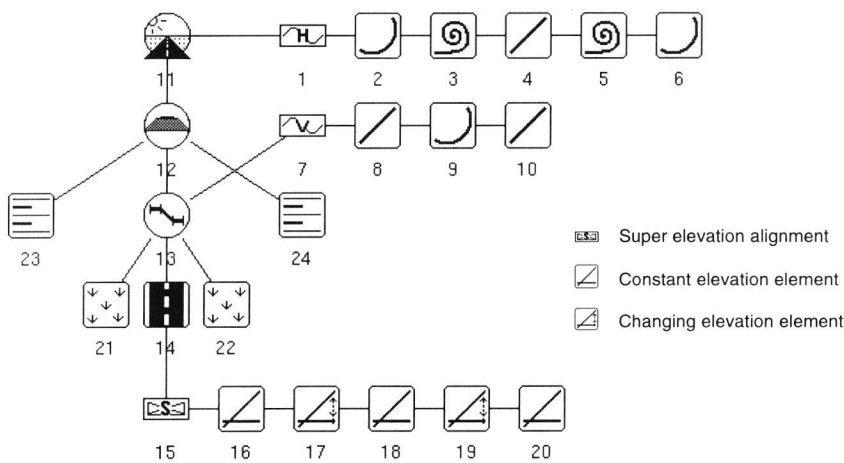


Figure 7.7.a Simple road structure with a super elevation alignment (15) attached to a carriage way (14).

The change in the transversal gradient is accomplished by a rotation⁵⁷ along a particular axis. The position of this axis of rotation is highly determined by the location where the vertical alignment lifts the road element complex. If this lifting point is:

- left of the referring road element primitive, then its left side coincides with the axis of rotation,
- right of the referring road element primitive, then its right side coincides with the axis of rotation,
- inside the referring road element primitive, there is a freedom of choice to settle this axis of rotation anywhere between the boundaries of the road element primitive. It is even conceivable to shift the location of the axis of rotation, as long as this shift takes place at a position where the transversal gradient = 0.⁵⁸

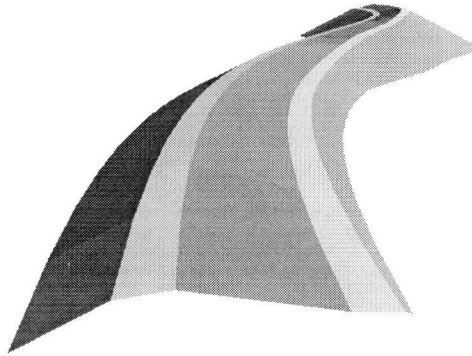


Figure 7.7.b Evaluation of example of figure 7.7.a

Width alignment

The width alignment controls the variation of the transversal size along a certain interval on the longitudinal axis. The change in the transversal size is accomplished by imposing a transformation at one side or distributed over both sides. The distribution of the transformation is highly determined by the location where the road section is anchored to the horizontal alignment. If this anchor point is:

⁵⁷) Strictly: not a real rotation because the projection of the road element primitive on the horizontal plane does not change.

⁵⁸) In this situation the vertical alignment curve logically coincides with the axis of rotation. An axis shift at a spot with a transversal gradient $\neq 0$ would introduce a discontinuity in the vertical alignment.

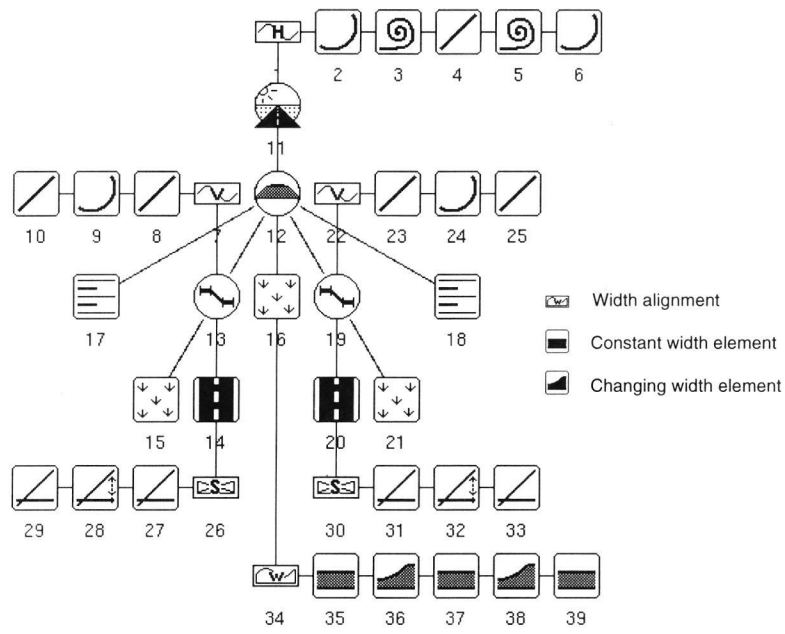


Figure 7.8.a Road structure with a width alignment (34) attached to a central reserve verge (16).

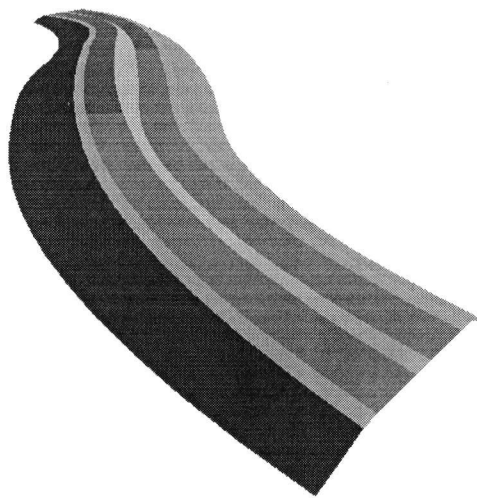


Figure 7.8.b Evaluation of example of figure 7.8.a.

- left of the referring road element primitive, then the transformation disseminates to the right,

- right of the referring road element primitive, then the transformation disseminates to the left,
- inside the referring road element primitive, then the transformation disseminates proportionally to both sides.

The first and second rule are logically forced to keep the horizontal alignment curve consistent, however, the third rule is a modelling choice to make a fluent transition between the first rule and the second.

7.3 Shape evaluation

The hierarchical structure of the RSMK and its clean separation of the general road structure and specific alignment data is particularly fit for evaluating its shape using the Space Deformation Tree (SDT) representation.

The SDT representation operates on two levels:

- At the lower level it uses two integrated graph types, space graphs and cell graphs, to represent respectively geometry and topology,
- At the higher level it uses a shape graph that can be evaluated into aggregations of space graphs and cell graphs.

The SDT-representation is described extensively in chapter 5, for the remainder of chapter it is sufficient to know that all shape nodes (primitive or complex) are contained in a separate modelling space. If a shape is incorporated in a shape complex the modelling space of the shape is embedded in the modelling space of the shape complex. The space embedding function may have three

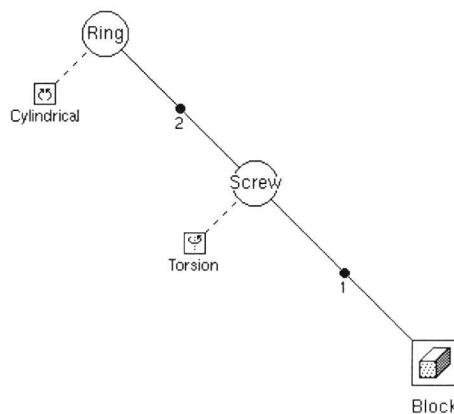


Figure 7.9 Shape graph of a ring of Moebius in three steps: block, screw, ring.

components: translation, rotation and deformation. For modelling ease at shape graph level only one deformation component can be specified. This deformation will affect all participating sub-shapes.

To demonstrate the relative ease the SDT representation at shape graph level offers to model complex shapes, the modelling steps to create a ring of Moebius is selected.

The shape graph (figure 7.9) is very simple: one shape primitive and two shape complex nodes. The dimensionality-three shape primitive is a block with length $2 \cdot \pi \cdot R$, with R the required radius of the ring. The modelling space that hosts this block is embedded in the modelling space of a shape complex (Screw) and twisted by a torsion deformation. The block in this space transforms into a screw with a 180° pitch. Finally the modelling space of the Screw shape complex is embedded in a modelling space of another shape complex (Ring) and bent through a cylindrical deformation. Figure 7.10 shows the resulting shapes of each modelling step.

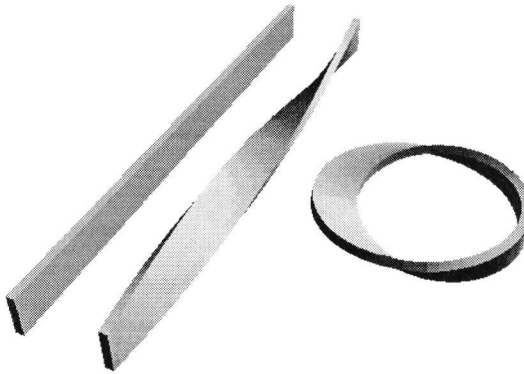


Figure 7.10 Block → screw → ring of Moebius.

7.3.1 Shape evaluation steps

Both data structures of RSMK and SDT are tree-oriented, which implies that the SDT structure can directly be raised in a one-pass traversal (depth first) from the RSMK product structure.

The shape evaluation process has the following steps:

- assemblage of a shape graph.
- assemblage of deformation components.
- generation of the more explicit space and cell graphs (geometry, topology).

- back-end generation of various graphical formats or application programming interfaces (API's), e.g. GKS, OpenGL, DXF and VRML.

The last two steps are standard and certainly not specific for roads, hence in the next sections only the first two steps will be discussed.

7.3.2 Assemblage of the shape graph

Road element primitive shape graph

In road modelling it is common practice to represent 3D models with surface modelling. Therefore, each road element primitive (carriage way, verge or slope) will be represented in the shape graph by a rectangular primitive. The length is determined from the longitudinal size of the road section in which this road element primitive participates. The width is always set to 1.0 and, in case of a constant width alignment, afterwards stretched to the appropriate value by a linear scaling deformation⁵⁹. The plug-in level of the width deformation is the shape complex node that directly embeds the rectangle primitive. The next shape complex node is the plug-in level for the transversal gradient deformation. Thus, the graph structure of the first three nodes, starting from a leaf in the direction of the root, is always the same (figure 7.11).

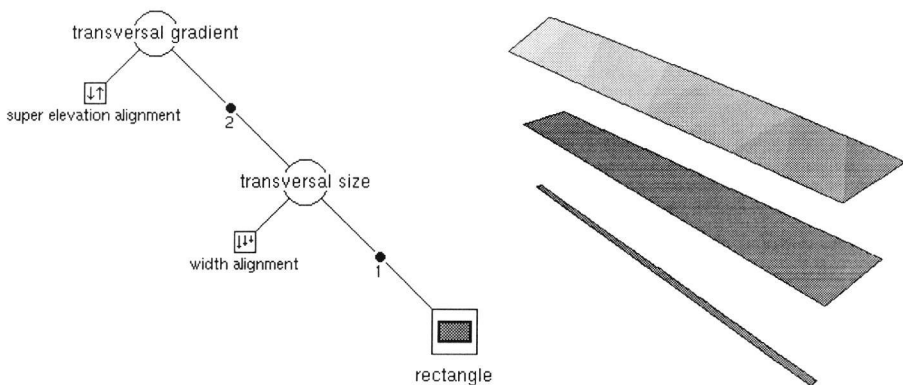


Figure 7.11 First three nodes in the shape tree starting from a leaf node: rectangle → application of the width deformation → application of the super elevation deformation.

⁵⁹) Of course, in the case of a constant width this value could directly be used to set the width parameter of the rectangle, but now the structure does not change if the constant width is changed into a variable width.

Road element complex shape graph

The next step is to combine a set of road element primitives into a road element complex. The objective is to create a continuous surface without any gaps, cracks or overlaps. To realise this aim the spatial effects of width alignments and super elevation alignments are propagated to neighbouring elements. A width alignment imposes a transversal shift, while a super elevation alignment imposes a vertical shift. These transversal and vertical shifts result into two extra deformations on the neighbouring branches of the shape tree. The direction of these shift propagations is determined by the position of the road element primitive with regard to the horizontal alignment (transversal shifting) or vertical alignment (vertical shifting) anchor points:

- an anchor point to the left propagates the shift to the right,
- an anchor point to the right propagates the shift to the left,
- if an anchor point is embedded in the road element primitive itself the RSMK

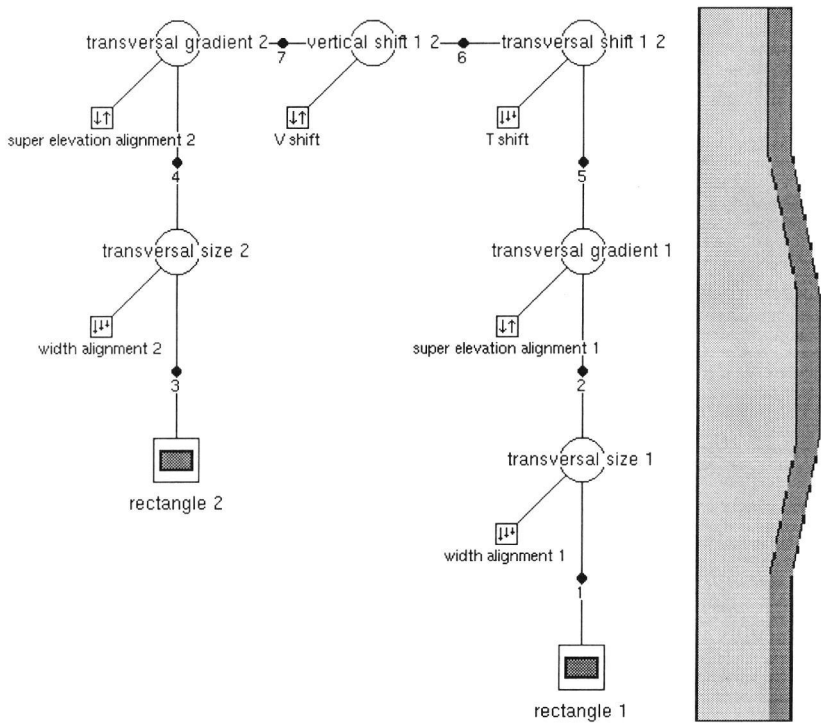


Figure 7.12 Coupling of two road element primitive shape representations. Here the spatial influence of the width alignment and super elevation alignment deformations operating on rectangle 2 are propagated to the rectangle 1 branch through two shift deforming shape nodes: transversal shift 1-2 and vertical shift 1-2.

prescribes a proportional distribution of the shift to both sides.

Taking into account two anchors at the same time imposes a problem to structure the shape tree. Therefore within the scope of a road element complex the vertical alignment anchor acts also as horizontal alignment anchor. This deviation is compensated when the shape sub-trees of the road element complexes are aggregated into the shape tree of a road section.

The next procedure creates a shape sub-tree to represent a road element complex:

1. Determine the anchor element of the vertical alignment, i.e. which road element primitive embeds the vertical alignment anchor point (or which road element primitive is left bounded by the vertical alignment curve).
2. Take the right most primitive and work from right to left until the anchor element.
3. Take the left most primitive and work from left to right until the anchor element.
4. Use the anchor element as root element and attach both branches to it.
5. Create a root shape complex node and attach the vertical alignment deformation to it.

Road section shape graph

Analogous to a road element complex the objective for assembling a road section shape graph is to create a continuous surface without any gaps, cracks or overlaps. Two important topics must be dealt with at this level:

- combining the participating road element complexes (each may be independently controlled by a vertical alignment) and the road element primitives that act as transition elements between each two neighbouring road element complexes.
- aligning the road section shape representation to the horizontal alignment anchor point.

The deformations, that are needed to fit in the shape representation of a transition element, are derived deformations. They are dependent from the shape evaluation results from the adjacent road element complexes. To determine these derived deformations partial shape evaluations are performed on the neighbouring shape sub-trees to generate the actual displacements. Then the fitting deformations can be composed either width determined with derived transversal gradients or transversal gradient determined with derived widths.

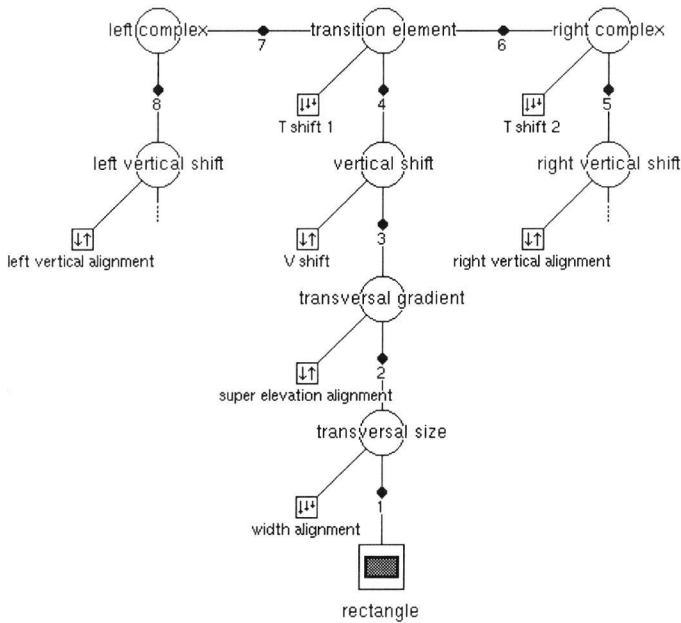


Figure 7.13 Coupling of two road element complex shape representations by a bridging road element primitive as transition element. The road element complex shape sub-trees have been left out in this diagram.

The next procedure creates a shape sub-tree to represent a road section:

1. Determine the anchor element of the horizontal alignment, i.e. which road element embeds the horizontal alignment anchor point (or which road element is left bounded by the horizontal alignment curve).
2. Take the right most primitive and work from right to left until the anchor element.
3. Take the left most primitive and work from left to right until the anchor element.
4. Use the anchor element as root element and attach both branches to it.

Road shape graph

Compared to the composition of the shape sub-tree assemblies the final road shape graph assemblage is rather trivial. The next procedure creates a shape tree to represent a road:

- combine sequentially the participating road sections (each may be independently aligned to its horizontal alignment anchor point) so that they are placed one after the other without gaps or overlaps.

- finally, create a root shape complex node and attach the horizontal alignment deformation to it.

7.3.3 Assemblage of complex deformation components

The deformations that are applied to raise the shape tree, as discussed in the previous section, may have a simple type or a complex type. In a simple deformation a single deformation function is defined for the complete embedded space domain. In a complex deformation the embedded space domain is divided into several disjoint subspaces each behaving according to a separate deformation function (and translation and rotation components). Points outside any bounding box are not supposed to experience any deformation. In a practical implementation the subspaces will be represented by bounding boxes. Obviously, much attention is needed to secure the continuity when traversing from one deformation zone to the next.

The deformation zones needed to represent the RSMK alignments all resemble a straight segmented tunnel where each tunnel segment handles an element from the corresponding alignment. A nice feature from this approach is compositional independence between the deformation complexes that are encountered during a shape evaluation. There is no need to ‘harmonise’ the different alignment decompositions, hence they may be changed at random while the spatial integrity is guaranteed.

Deformation complexes are assembled for width alignments, super elevation alignments, vertical alignments and horizontal alignments, besides they are

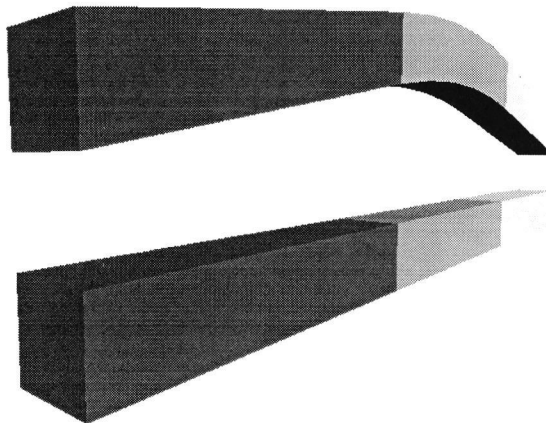


Figure 7.14 Deformation zones of a deformation complex to implement a vertical alignment, before (below) and after (above) the deformation is applied.

generated to support 'derived alignments' for transversal and vertical shifts.

The scope of a deformation complex may vary between:

- the modelling space of a single road element primitive shape representation,
- the modelling space of a branch in the shape tree (e.g. shift everything left of this road element) or
- the complete shape tree by plugging in the horizontal alignment in the root shape complex.

Besides, the same deformation complex can be shared by multiple shape complexes, e.g. to mirror the same width alignment at both sides of the horizontal alignment curve or by reusing the vertical alignment for two carriage ways.

7.4 Conclusions

The RSMK is not just an exercise on paper. The complete model is developed and implemented with the product modelling environment tool PMshell [Luijten 1996]. The model implementation is embedded in a tool called PoC (from Proof of Concept) that not only supports the direct instancing (or importing/exporting a STEP file) of a road data model that satisfies the RSMK definition, but also the shape evaluations to derive an explicit 3D model from it, according to the principles described in the previous sections. With this tool all examples in this chapter were entered and evaluated into 3D models and afterwards rendered into shaded images.

The next conclusions can be drawn from this case:

- **Product modelling**
The roads case shows that product modelling structures not only fit assembly/parts like products with clear-cut interfaces, other more monolithic products, like roads, can also be addressed. By introducing structural hierarchies in road modelling there is no need anymore to start all over again after for instance a major modification in an alignment or a cross section. An important increase of reusability is the direct result of this approach. By separating the one-of-a-kind characteristics (read: alignment geometries) from the remainder of the road model (read: its product structure) it is possible to introduce parametric part libraries (e.g. standard road features as access roads, splitting and joining of carriageways/lanes, standard intersections and so on) ready for use in road modelling.
The Road Shape Model Kernel uses only a limited set of structuring mechanisms, essentially a decomposition of occurrence objects. Although this

may offer a satisfactory model, certain questions, for instance with regard to the continuity over two adjacent road sections, are hard to check. Besides, global changes, e.g. to change the gradients of all slopes, must be applied to each separate occurrence of that particular object type.

- Shape modelling

The Space Deformation Tree representation has demonstrated in this project its competence by the shape evaluation of the complete Road Shape Model Kernel test suite. Its structure integrates in a natural way various co-ordinate systems (global versus local and Euclidean versus non-Euclidean), which offers the opportunity to choose the most appropriate co-ordinate system for each situation. The SDT representation can also be applied to guarantee all kinds of continuity rules both in longitudinal and transversal direction.

- Integration of product modelling and shape modelling

Because of the full parametric shape description of the Road Shape Model Kernel the generated SDT representation is automatically in line with the product model structure.

8

Case 2: Viaducts

A viaduct is a technical solution for a crossing of two or more roads. Therefore it is obvious that a product model for viaducts should be related to a general road model (like for instance the model discussed in the previous chapter). This chapter considers two projects: the Rijkswaterstaat ViaDesign project and the ESPRIT GEM project that jointly span both the load support systems as the road carrier functions of a viaduct.

8.1 Introduction

The first case was devoted to the modelling domain of shape definition for highway design, which was formally recorded in the information model: Road Shape Model Kernel (RSMK). Soon it was recognised that the same approach could be used for special structures in a highway, like tunnels, viaducts and fly-overs. These special structures should also conform to the globally designed highway geometry, i.e. to the horizontal and vertical alignment definitions. In addition, the functional part of the road (the actual traffic carriers: the carriage-ways) should also stay in place. The differences have primarily to do with the structural mechanics and the applied building materials (soil versus concrete or steel), and the fact that often two or more (in)dependent roads are involved.

The results of this Ph.D. research have been brought in into two projects:

- the development of an information model to record the support structure definition of a viaduct. This model was the result of a work item of the ViaDesign project, which aim is to realise an intelligent environment for the early shape design of viaducts. ViaDesign will be developed by the Dutch ministry of Public Works (Rijkswaterstaat) and TNO.
- the prototype development of a parametric viaduct modeller to demonstrate the data exchange between Computer Aided Design and Engineering Analysis. The demo-project is a deliverable of the ESPRIT GEM (Generic Engineering-analysis Model) project.

8.2 Viaduct support structure definition

8.2.1 The relationship between road design and viaduct design

Obviously, there is a close relationship between road design and viaduct design: a viaduct is a technical solution for a crossing of two or more roads. This dependence is also reflected in the design process: basically the road design comes first resulting in various crossings with other roads. These other roads can either be existing roads or roads that are also in the design stage. A viaduct is a possible technical solution for a crossing: taking one road (the top road) over one or more other roads (the bottom roads). Instead of one road (as discussed in the previous chapter) a viaduct has typically to deal with multiple roads.

A viaduct can be decomposed into the road carrying structure (the viaduct deck) and the support structure: two end support structures (the abutments) and zero or more intermediate support structures (the pillars). Generally, the road carrying structure will follow the alignment of the top road while the support structure is constrained by top road *and* bottom roads alignments. This section will focus primarily on the support structure while the next section will consider the road carrying structure.

In an integrated information model environment a road design model and a viaduct design model could be related as follows (figure 8.1):

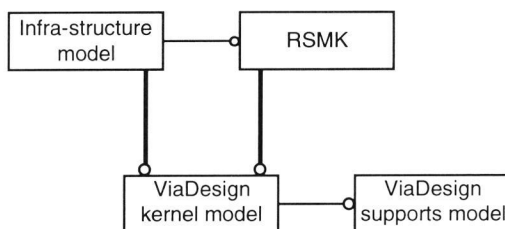


Figure 8.1 Model framework for road design and viaduct design. The fat solid lines denote inheritance relationships: the viaduct model is a specialised infra-structure model with some specialised road features.

The Road Shape Model Kernel (RSMK) describes elementary pieces of a road (including the alignments) without intersections. An infra-structure model (not yet developed) could aggregate these singular pieces to a road network including crossings. The viaduct model (ViaDesign kernel model) should inherit from both models: the crossing concept from the infra-structure model and the individual road descriptions from the road model. The viaduct supports model

follows from the viaduct decomposition into a road carrying structure and the support structure.

Because this model framework is not (yet) available, the viaduct supports model uses the RSMK as its parent model while a few entities are added (*Road-InfraStructure* and *Crossing*), that should be inherited from an infra-structure model.

8.2.2 Viaduct support structure model features

The viaduct supports model has incorporated several modelling features to support the relationships between road design and viaduct design, as discussed in the previous section, and the design of viaduct support structures. To mention the most significant features:

- *viaduct level - deck level*

A primary distinction is made between the viaduct as a whole and the viaduct deck as a part. This distinction is appropriate to map these concepts as specialisations of the RSMK entities *RoadSection* and *RoadElementComplex* respectively. This means that a viaduct with more than one deck (with independent vertical alignment specifications) is already included in the model by defining just these two inheritance clauses.

The support structures themselves are defined at viaduct level, while the support interfaces are defined at viaduct deck level. The reason is that the number of supports will not multiply with more than one viaduct deck⁶⁰.

- *shape contours*

The viaduct supports model does not facilitate detailed shape modelling of the support structures. Instead a kind of bounding box is established to define a contour that the final shape is not supposed to exceed. The rationale is that especially the domain of possible shapes for pillar structures is too vast for parameterisation.

⁶⁰ If this is not the case this model forces to create more than one viaduct, which is probably what is actually meant.

- *support structure location (absolute/relative)*

The location of a support structure (*SupportPoint*) can be defined absolutely (with static x and y co-ordinates) or relatively (with regard to the road alignment). In the viaduct supports model a *SupportPoint* is located relatively in the longitudinal and transversal co-ordinates of the upper road. However, if the SDT representation is generated both locations are available, depending the node the query is issued (See figure 8.2).

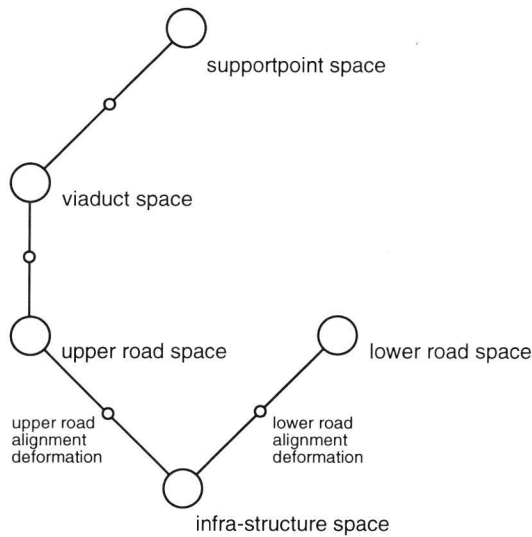


Figure 8.2 Part of the SDT space graph for a viaduct support structure. Location queries from the nodes 'upper road space', 'viaduct space' and 'supportpoint space' will all result in 'upper road alignment' co-ordinates, while a query from the 'infra-structure space' node returns absolute x/y co-ordinates. The support structure location relative to the 'lower road space' node can be derived (if the location is in the "neighbourhood" of the longitudinal axis).

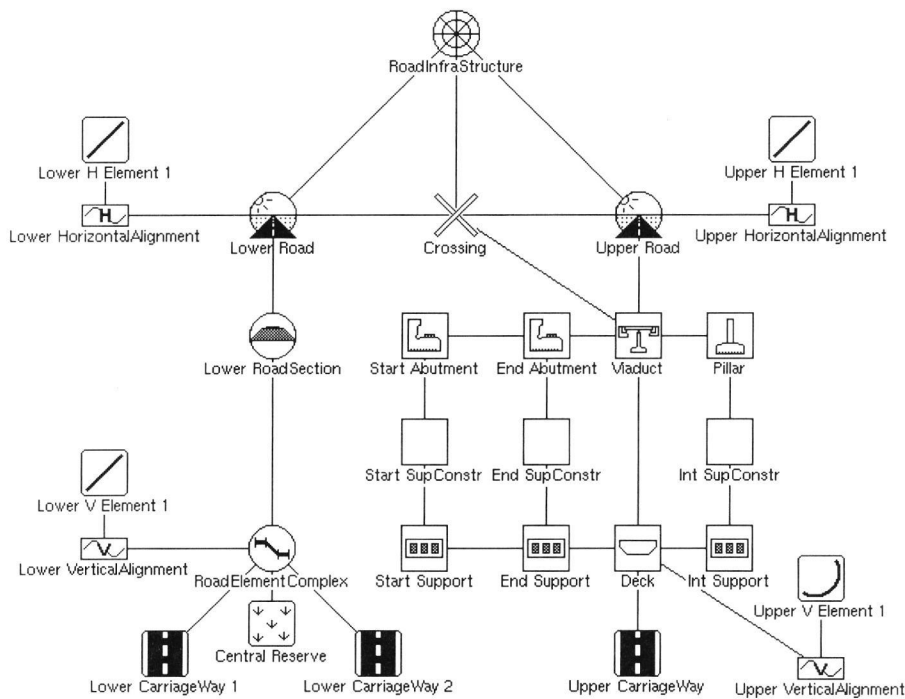


Figure 8.3 Product structure diagram of a viaduct embedded in limited road infrastructure that consists of two roads with a common crossing.

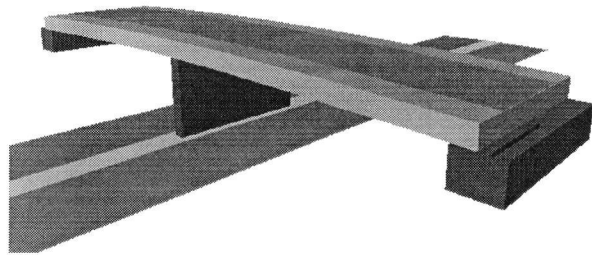


Figure 8.4 3D solid geometry that is rendered from the product structure as described in figure 8.3.

- *support structure shape (top, bottom, transition)*
Basically support structures have two locations: a bottom location, with regard to the viaduct as a whole, and a top location, with regard to the viaduct deck. This offers a lot of freedom, for instance slanted and/or twisted structures.

8.2.3 Shape evaluation

Shape evaluation was implemented in a rapid prototype modeller as a kind of proof of concept. Because the model defines primarily locations and bounding boxes, the rendered geometry makes a rather crude impression. However, all features mentioned in the previous section could be demonstrated.

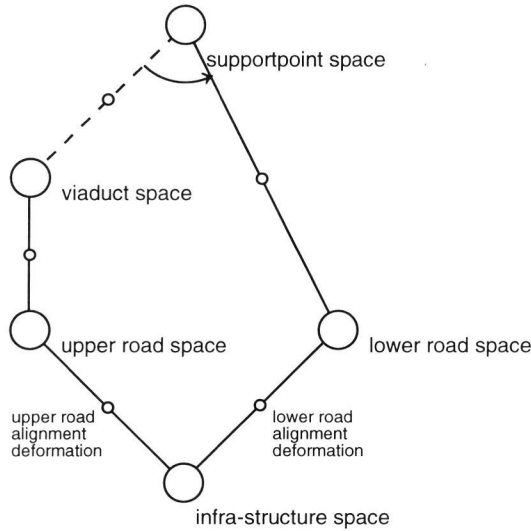


Figure 8.5 Rewiring operation of the Space Deformation Tree to align a support structure with the lower road deformation.

One complexity arose because of the fact that the algorithm for the SDT representation generation follows the product decomposition tree. This means that for this model the support structures all belong to the upper road branch. If a model instantiation prescribes that a support structure must be aligned to the horizontal alignment of the lower road, the shape nodes in the SDT representation of this particular support structure had to be disconnected and re-attached to the lower road branch (figure 8.5).

8.3 Parametric viaduct modeller

8.3.1 The ESPRIT GEM project and the Fly-over demo

The prime objective of the ESPRIT Generic Engineering-analysis Model (GEM) project [Böhms, Rousset, Miles, Leal and Helpenstein 1995] is the development of a neutral and STEP compliant information model to communicate engineering-analysis data between (FEM) analysis applications, FEM-pre-processors, FEM-post-processors and CAD-systems. Besides the main demonstrators Fiat (fog lamp) and Dornier (satellite), a limited TNO-only demonstration has been prepared in co-operation with Rijkswaterstaat: the Fly-over demonstration. A prototype parameterised viaduct modelling application is developed to demonstrate especially the data transfer from a CAD-system to a GEM-database. After an automatic meshing operation within the FEMGV package a stress analysis with the TNO DIANA package should complete the demonstration.

8.3.2 Viaduct parametrics

The viaduct modeller is developed around a small viaduct product type model, that uses the Road Shape Model Kernel (RSMK) [Willems 1995] as its parent model. Of course, all alignment definition entities are already available from the RSMK. The modelling perspective is to mark a viaduct as a specialisation of a *road section* and a viaduct deck as a specialisation of a *road element complex*.⁶¹ The viaduct model itself deals mostly with the decomposition of a viaduct deck into a set of field zones and support zones, and the specification of the location and behaviour of the actual physical interfaces of the support structures. The support structures themselves are considered out of scope here⁶².

The viaduct modeller uses a fully parametric approach, i.e. the resulting shape is the consequence of the combined values of a limited set of parameters. Although limited, the total number of parameter values for a particular viaduct is not fixed. Certain parameter value choices will generate additional parameters, while other parameter groups are aggregations (with often undefined cardinality) of more elementary parameter sets. For example, the complexity of an alignment specification can be broken down into its primitive alignment elements each fully defined by a few parameters.

The parameter set can be categorised in a number of groups:

- road geometry parameters

⁶¹ See chapter 7 for a explanation of these RSMK entities.

⁶² Integration with the previous support structure model could result in a kind of Viaduct Shape Model Kernel of course.

This parameter set is inherited from the road design parent model:

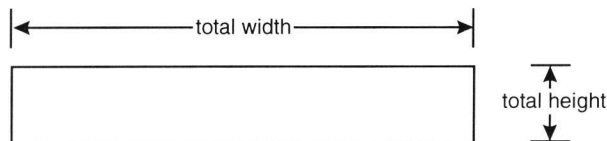
- horizontal alignment
- vertical alignment
- super-elevation
- road topology parameters

This parameter set is also inherited from the road design parent model however new cross-section primitives may be added. For instance, a road with two separate carriageways may be carried by two separate and structural independent viaduct decks. Still, the road design defines a single road. In this case the linking element (the intermediate verge or central reserve) is necessary to control the distance between the two carriageways but is not materialised in the actual viaduct realisation phase.

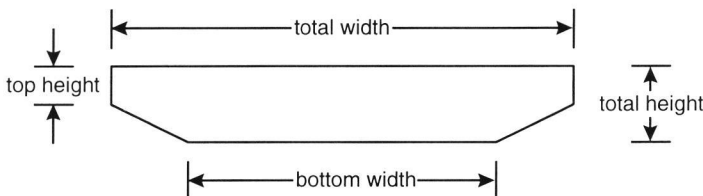
- general viaduct parameters
 - total length
 - number of intermediate supports
 - applied material properties (concrete)
- transversal parameters

For the demo three different and widely applied viaduct deck cross-sections are defined:

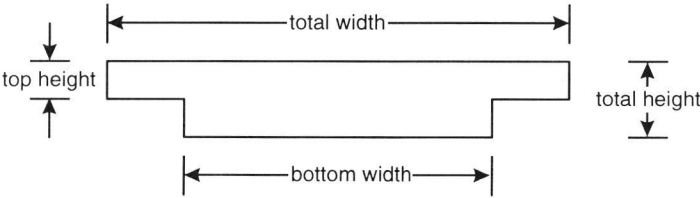
- rectangle



- trapezium



- T-profile



• longitudinal parameters

The length profile offers various parameters to control the variations of the construction height in the longitudinal direction. The viaduct modeller offers four choices mapping to the degree n of the polynomial function

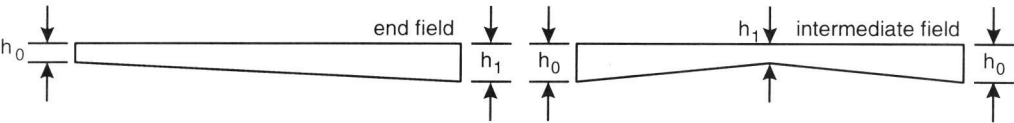
$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

i.e.,

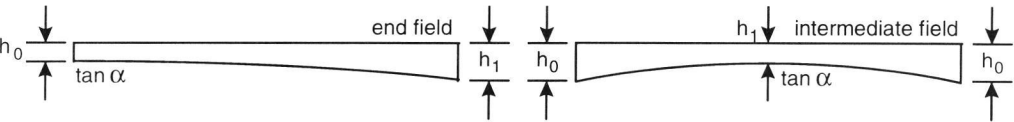
- $n = 0$, constant height function, one parameter: h ,



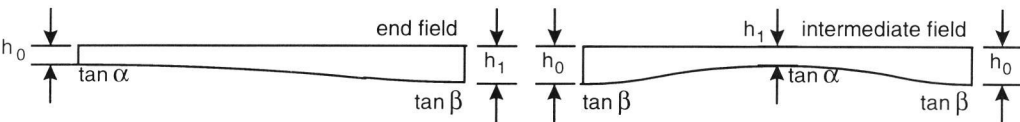
- $n = 1$, linear height function, two parameters: h_0 and h_1 ,



- $n = 2$, quadratic height function, three parameters: h_0 , h_1 and one gradient,



- $n = 3$, cubic height function, four parameters: h_0 , h_1 and two gradients.



The variation of the construction height in longitudinal direction may affect the variation of other parameters if there is a dependence relation

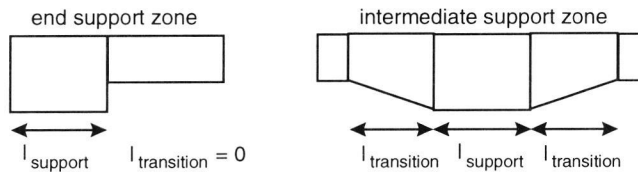
(constraint) between them. For example, in case of the trapezium deck cross-section it is common practice to keep the angle of the slanted sides constant over the length of the viaduct (for ease of straight reinforcement bars, flat concrete shuttering or simply for aesthetic reasons). To realise this constraint the bottom width parameter value should be directly coupled to the total height parameter value at every longitudinal co-ordinate:

$$w_{\text{bottom}}(x) = (w_{\text{bottom}}(x_0) / h_{\text{total}}(x_0)) h_{\text{total}}(x)$$

where x_0 is a longitudinal co-ordinate somewhere in the definition interval.

Besides construction height variation in the field zones of the viaduct deck some parameters describe the construction height in the support zones:

- the length of the constant construction height directly above a support (end support or intermediate support),
- the length of the (linear) transition zone from support zone to field zone. A zero valued transition zone length may result in a discontinuity of the construction height.



A parameter group is specified in its local context, i.e., without the interference of other parameter groups. This means that for instance the longitudinal profile is specified as if the top plane of the viaduct deck is straight and flat. The effects of the horizontal, vertical and super-elevation alignments are only applied when the total shape tree is evaluated. The consequence is that the total number of necessary parameter values that must be specified for an average viaduct is still very limited. More over, a late change in the alignment specification can be applied without 'starting all over again'.

8.3.3 Shape evaluation and B-rep generation

The Space Deformation Tree representation (SDT-rep) evaluation follows essentially the same strategy as in the first (roads) case. However, there are a few differences:

- in the roads case only the top surface is generated: the result is basically a surface model. Here are mostly solid elements involved. This is more complicated because there are much more constraints to be fulfilled. For in-

stance, if the carriageway is slanted (has a transversal gradient), should the deck follow this geometry with

- a rotation (all points move radially around a rotation axis over a certain angle), or with
- a skewing (all points move around a rotation axis over a certain angle parallel to the y axis, preserving the original x co-ordinate), or with
- a transformation from a rectangular contour to a trapezium contour (see figure 8.6)?



Figure 8.6 Various deck rotations/deformations to follow a transversal gradient of the carriageway.

- the support structure only partially follows the main road alignment geometry, i.e. x and y location follow mostly the horizontal alignment, however the vertical alignment affects only the location of the physical deck/support interfaces but not the z location of the support structures themselves.

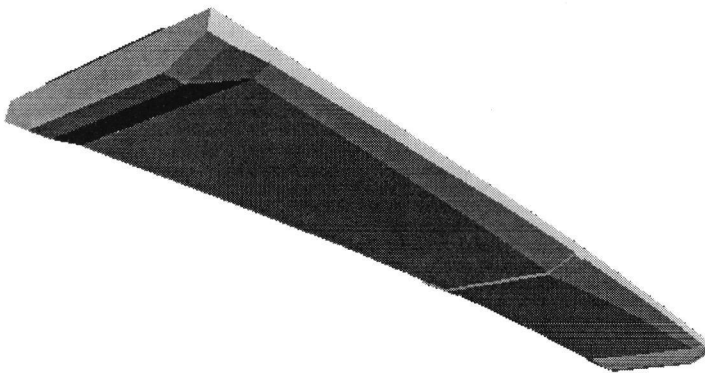


Figure 8.7 Shaded image rendered from a VRML representation of a viaduct deck that was generated by the GEM Fly-over modeller.

Because the Fly-over demo of the GEM project focuses only on the deck structure (support structures are considered out of scope), this issue could be ignored here. See further the previous section especially dealing with support structure.

The shape description of the GEM core model adapts Part 42 of STEP [ISO/TC184 1994d] or more accurate, a sub-set of this integrated resource model Application Interpreted Construct 514: Advanced Boundary representation (AIC 514). In this case it proved relatively straightforward to convert the SDT-rep into this format, nevertheless, this should be tested much more thoroughly to draw any conclusions.

8.4 Conclusions

The next conclusions can be drawn from this case:

- Product modelling

The viaduct case shows that parametric modelling, which is closely associated with product modelling, is a very powerful method to speed up the design process of 80 to 90 percent of all viaducts. Until now, the rather complex and unique individual shape of each viaduct occurrence prevented a parameterisation with sufficient flexibility.

The case also shows that product modelling offers the opportunity to combine the various components of a large project that traditionally are handled separately. Complete units, for instance all viaducts and road sections of a large highway intersection project, can be plugged in a global infrastructure level. A more smooth integration of the various parts will be the result.

- Shape modelling

The SDT-representation offers the possibility to model viaduct and road alignment separately, i.e. the viaduct parameterisation records a straight and flat structure, while the alignment specs mould it later to the appropriate shape.

The various coupled co-ordinate systems support for instance that one can specify the location of the support structures in global independent co-ordinates, while the position of the crash-barrier on the viaduct deck can be specified in local iso-parametric co-ordinates.

- Integration of product modelling and shape modelling

See corresponding item of the roads case.

9

Summary, Conclusions and Recommendations

9.1 Summary

9.1.1 The problem

The Construction Industry is lagging behind in the uptake of state-of-the-art Information Technologies in its design and realisation processes. The one-of-a-kind character of its products, the site-bounded nature of its manufacturing process, the highly flexible but temporary characteristics of its organisation and production process, the complex and interrelated topologies and the unique individual shapes of its products are the critical factors that explain the hesitant admission of technological innovations in the Construction Industry. In the future the problems faced by the industry will become worse. Stronger demands on safety, environmental pollution, energy savings and end-user satisfaction, come together with increased demands on competitiveness and reduction of realisation times. The design/realisation process should display enough flexibility to allow late (sometimes very late) changes in the product requirements specification. Such a flexible design process is in conflict with the dominant start-from-scratch practise of the Construction Industry

Application of information technologies, like Product Data Technology (PDT), Computer Integrated Construction (CIC), Concurrent and Co-operative Design/Engineering and Realisation are pursued as potential solutions by other sectors of industry that face similar problems. Compared with those other branches of industry the Construction Industry stays well behind in employing contemporary information technology in the realisation of a project. Although many activities are supported by computers, the communication infra-structure in a building project is still inadequate to harvest the true benefits of a computerised information system. Low level information interfaces are short-lived under these conditions because they lack the necessary flexibility. A high quality communication infra-structure is in principle capable to raise the level of efficiency of the production process and the quality of its products. A higher level

of meaning or semantics seems to be the key factor for improving the communication between computers and between computers and humans in the Construction Industry.

9.1.2 The proposed solution

Representation

Traditionally, the Construction Industry uses technical drawings as the prime representation of building products. The drawing representation, in principle aimed at human interpretation, is much less appropriate for state-of-the-art machine interpretation.

To raise the level of meaning and, more in general, the quality of the information technology infra-structure the representation forms, that are communicated between computers, should conform to the following requirements:

- preferably an implicit representation, from which other more explicit representations can be derived,
- a machine-interpretable representation, to realise genuine integration of computer-supported processes, and
- a shape independent representation structure, to support multiple shape models for the same object and to offer a backbone for all other, often shape independent, properties.

A product model (that conforms to a well defined, flexible and semantically rich conceptual structure) is a representation form that is capable to meet all these requirements. A product model is an information model that is able to store, in a format that is neutral with regard to any specific application, all the information of a product during one or more life cycle stages.

Product modelling structures

Product modelling is only feasible if the applied models conform to a well defined, flexible and semantically rich conceptual structure. Structuring mechanisms as scope, specification level, decomposition, concretisation, modularity, connectivity and life cycle stage are necessary ingredients for such a super-structure. On the other hand this super-structure should not impose unnecessary constraints on the modelling freedom of product type model developers or force them to use mechanism they are not interested in. Of course, this last observation is also a very important prerequisite for a product and process reference model to be admitted by the modelling community.

A flexible product model structure is also a necessity for proposing the appropriate slots for plugging in the various shape models that can be attached to the same product model. If the product type model is well structured each shape model must have a unique slot using its 'co-ordinates' in several (or maybe) all available dimensions. Modularity in shape modelling may help to break down a complete but large shape model (that frequently will address probably several slots) and distribute it in a modular form so that modules and slots form one-to-one relationships.

An example product and process reference model is proposed that is able to support a large set of structuring mechanisms. The addressed modelling domain is stretched out by five structuring mechanisms called modelling dimensions. These dimensions are considered orthogonal, hence, entities can be formed by combining up to five independent entities from each of the five modelling axes. Although a product and process reference model like this can be applied for direct instantiating, the semantic content is evidently too general to be of much significance for this purpose. The justification for the existence of such a reference model is to ease the integration and the inter-operability of product and process models that are developed for a particular branch of industry. The idea is to develop a layered hierarchy of models ranging from broad scoped high level models to narrow scoped low level models all conforming to a common product and process reference model. In such a framework each slot should contain a model of a well-defined and limited scope that could act as a parent model for more specific models or as a server model for other client models. The conformance of the applied structuring mechanisms in each model helps to achieve a kind of plug-in architecture encouraging the reusability of the well-developed models and the replacement of less-achieving models.

Modelling complex unique individual shapes

Shape representation is traditionally the domain of geometric modelling and computer graphics. Over the years the data structures in geometric modelling have evolved to fulfil requirements in the area of optimal evaluation speed or robust geometric operations support. Product modelling imposes different structural needs, which cannot always be satisfied by the present day shape representation methods. Important structural needs are for instance:

- the ability to follow the basic modelling structures of decomposition and connectivity,
- support for parametric (i.e. reusable) shape definition,
- support for modularity and

- adequate freedom to make spatial claims for a specific node in the product structure.

A new shape representation is introduced that is developed from the product modelling perspective: the Space Deformation Tree (SDT) representation. The Space Deformation Tree representation has several characteristics that make it particular suitable for association with product models:

- The geometry part (the actual space deformation tree) offers the opportunity to define a shape object as a parametric library part in a neutral undeformed space. Plugging the root space of such a detached shape into the actual space graph of a shape model will mould it according to the actual valid geometry. This technique will increase the potential domain for parametric shape objects (and therefore the possibilities of reusing previous designed objects) considerably.
- The topology part satisfies effectively non-manifold topology, which is a prerequisite for implementing explicit connectivity and reference geometry representation.
- The shape part maps nicely on various decomposition (type/occurrence) strategies.
- A space graph / cell graph combination can be considered as a sub-model (modularity) and handled as a macro-object at shape level. Integration (merging) of space graphs or cell graphs offers a flexible tool to assemble new shape objects out of a limited shape primitive set.
- Many options are offered to record spatial claims for a particular node in a product model. These claims may be disjunct or overlapping. The same facilities are used to record spatial connectivity.

Integration of product modelling and shape modelling

A complex product can be represented by many different shape models, however there should exist only one unique product model. The product model will normally show a hierarchical structure: global information near the root and detailed information near the leaves. This hierarchical product information structure is also convenient to locate the different shape models, distributing their sets of shape elements over the leaves (and sometimes also over the intermediate nodes) of the product tree. Traditionally hardly any constraints are imposed to these shape models, which reduce their interrelations to the single fact that they are attached to the same product model. This situation with numerous independently defined shape models is unavoidable in an information

system with so-called islands of automation, however it is unacceptable for an integrated information system.

Criteria must be stated to help to compare two shape models and to decide if they are consistent with each other and their common product model structure:

- *accuracy*
By stating that the point set of the maximum shape of the model with the larger accuracy value must be the super set of the point set of the maximum shape of the model with the smaller accuracy value, a natural kind of a global / detail hierarchy is introduced.
- *harmonised decomposition*
A shape model can also be structured along a decomposition tree. By coupling the shape decomposition to the product structure decomposition, the shape model offers as many referencable entries as there are nodes in the tree, instead of only one entry: the root node. If all shape models are aligned following the same decomposition an important requirement for comparison is fulfilled.
- *explicit connectivity*
Explicit connectivity is another criterion to compare the consistency of shape models.

9.1.3 Evaluation of the case studies

Two realistic case studies have been used to demonstrate the usefulness of the concepts that are explained in the chapters 4 (Product Modelling Structures), 5 (Shape Modelling) and 6 (Integration of Product Modelling and Shape Modelling). The two case studies are realistic because they were selected from the common set of projects TNO was executing for the Dutch government (Rijkswaterstaat) in parallel with this PhD study. However, these projects did not contain special objectives to act as proof of concept for the ideas discussed in this thesis. As a result most concepts could be demonstrated adequately but not all of them. Especially the integration part is insufficiently exposed.

Roads

Road design has always played an important role in product data technology. Yet, the development of an information model for roads that supports a higher level of semantics than a bare minimum of explicit geometry (e.g. a labelled set of 3D strings) is not a trivial task.

- *Product modelling*

The roads case shows that product modelling structures not only fit assembly/parts-like products with clear-cut interfaces, also other more monolithic products, like roads, can be addressed. By introducing structural hierarchies in road modelling, there is no need anymore to start all over again after for instance a major modification in an alignment or a cross section. An important increase of reusability is the direct result of this approach. By separating the one-of-kind characteristics (read: alignment geometries) from the remainder of the road model (read: its product structure), it is possible to introduce parametric part libraries (e.g. standard road features as access roads, splitting and joining of carriageways/lanes, standard intersections and so on) ready for use in road modelling.

The Road Shape Model Kernel uses only a limited set of structuring mechanisms, essentially a decomposition of occurrence objects. Although this may offer a satisfactory model, certain questions are hard to check, for instance with regard to the continuity over two adjacent road sections. Besides, global changes (e.g. to change the gradients of all slopes) must be applied to each separate occurrence of that particular object type.

- *Shape modelling*

The Space Deformation Tree representation has demonstrated in this project its competence by the shape evaluation of the complete Road Shape Model Kernel test suite. Its structure integrates in a natural way various modelling spaces, which offers the opportunity to choose the most appropriate modelling space for each situation. The SDT representation can also be applied to guarantee plenty of continuity rules both in longitudinal and transversal direction.

- *Integration of product modelling and shape modelling*

Because of the full parametric shape description of the Road Shape Model Kernel the generated SDT representation is automatically in line with the product model structure.

Viaducts

A viaduct is a technical solution for a crossing of two or more roads. Therefore it is obvious that a product model for viaducts should be related to a general road model (like for instance the RSMK). Two projects are considered: the Rijkswaterstaat ViaDesign project and the ESPRIT GEM (Generic Engineering analysis Model) project that jointly span both the load support systems as the road carrier functions of a viaduct.

- *Product modelling*

The viaduct case shows that parametric modelling, which is closely associated with product modelling, is a very powerful method to speed up the design process of 80 to 90 percent of all viaducts significantly. Until now, the complex and unique individual shape of each viaduct occurrence prevented a parameterisation with sufficient flexibility.

The case also shows that product modelling offers the opportunity to combine the various components of a large project that traditionally were handled separately. Complete units, for instance all viaducts and road sections of a large highway intersection project, can be plugged in a global infrastructure level. A more smooth integration of the various parts will be the result.

- *Shape modelling*

The SDT-representation offers the possibility to model viaduct and road alignment separately, i.e. the viaduct parameterisation records a straight and flat structure, while the alignment specs mould it later to the appropriate shape.

The various coupled modelling spaces support for instance that one can specify the location of the support structures in global independent co-ordinates, while the position of the crash-barrier on the viaduct deck can be specified in local iso-parametric co-ordinates.

- *Integration of product modelling and shape modelling*

See corresponding item of the roads case.

9.2 Conclusions

In chapter 2 the following research questions are formulated.

9.2.1 Research question 1

1. *Which product and process representation is able to meet the requirements of the Construction Industry?*

The quality of the information system will be the key factor to bring the Construction Industry on a higher level of efficiency, i.e.:

- a more optimised design by raising the reusability of the design of particular building components,
- a higher deployment of dedicated autonomous machines in the manufacturing process,
- a more smooth transition from design stage to manufacturing stage,
- a better integration of sub-contractor activities.

The answer is that product modelling in principle is able to meet these requirements. However, the choice for product (and process) modelling is primarily founded on the conclusion that other candidate (shape related) representations have manifest disadvantages. For product modelling to fulfil its promise, i.e. an application independent data structure that is able to store and retrieve all the information of a certain product during its complete life cycle, it will be obvious that especially the structuring part turns out to be crucial. Therefore, in chapter 3 research question 1 is further detailed by questions 1.1 and 1.2:

1.1 Which structuring mechanisms can be employed to organise a product (and process) model?

The set of structuring mechanisms discussed in this chapter 4 can be encountered in various combinations and interpretations in numerous projects that have addressed this research field. Without claiming completeness this set of structuring mechanisms should belong to the core of available product and process modelling features.

1.2 How can these structuring mechanisms be integrated in one super-structure?

The example product and process model of chapter 4 shows the advantages of combining various structuring mechanisms in orthogonal modelling dimensions. Thus, a rich set of latent modelling combinations can be spanned leaving the model developer enough freedom to choose his distinct mixture of modelling constructs.

9.2.2 Research question 2

2. Which shape representation is able to meet the requirements of the Construction Industry?

Chapter 4 (Product modelling structures) has selected a set of modelling structures to support the development and usage of flexible product models. Data structures for shape modelling are typically developed to satisfy different requirements and usually do not display this flexibility. For shape models to integrate seamless in this modelling environment additional requirements must be stated, especially with respect to modularity. This leads to the following, more detailed, research question:

2.1 *Which shape representation is able to meet the necessary requirements with respect to flexibility (modularity) to integrate well with the modelling structures of a product model?*

Present day shape representations are optimised to satisfy precisely this single function: representing shape. Requirements for reusability and flexibility demand a more modular or object oriented approach.

The Space Deformation Tree representation incorporates a number of features that makes it more fit to fulfil these additional requirements at least for an important product domain in the Construction Industry: linear infra-structural objects (roads, railways, tunnels, channels, viaducts, bridges, dikes and their combinations.).

9.2.3 Research question 3

How to integrate both (product/process and shape) representations?

This question determines an important additional requirement to both the product (and process) representation and the shape representation to support interoperability between the functional properties and the spatial properties.

Various levels of integration can be distinguished. It is relatively easy to plug in one or more independent shape models to the same product model structure. Of course, this is a very shallow integration, however, if it is sufficient there is no reason to go further. In a more sophisticated approach the shape models could follow the decomposition structure of the product model and finally also the connectivity structure. As is outlined in chapter 6 criteria for comparison of shape models are necessary to control the ultimate integration of product modelling and shape modelling. Especially this integration issue needs more research and in particular more implementation experience in real projects then could be realised for this dissertation.

9.3 Recommendations

The following recommendations do not all result directly from this PhD research, yet they come forth from observations of the developments in information technology and product data technology over the last ten years.

- Standardisation is crucial for product data technology. Nevertheless, standards should be a means instead of a target. The employment of a standard like ISO 10303 (STEP) would spread much faster if simultaneously useful STEP-based products were developed. To compare: what would be the impact of standardisation of the programming language Java without the simultaneous development and availability of compilers, libraries, development environments, and so on? A similar approach for STEP could be the availability in public domain of:
 - certified EXPRESS tools,
 - STEP physical file I/O class library,
 - SDAI access class library,
 - EXPRESS/C++ or EXPRESS/Java generator,
 - EXPRESS/SQL converter,
 - documentation generator and
 - something like 'STEP Foundation Classes'?

This list sketches only some first ideas that could easily be extended. Yet, the general idea is: sell the standard via useful products that incorporate the standard.

- Although this thesis stresses the importance of integrated product and process modelling, the elaboration of the various issues is in terms of conventional static data modelling techniques. The underlying paradigm is a clear distinction between the data and the software that operates the data. Until now the world of product data modelling and the world of object oriented software development did not have much overlap. However, technologies as reusable software components (ActiveX, Java-Beans), distributed network communication (CORBA, IIOP, Java RMI) and platform independent software (Java) offer great opportunities to create highly dynamic models. This could change the 'snap-shot' data model exchange technology into a situation, where all partners share the same 'ever-green' model.
- Maybe the already mentioned reusable software component technology could turn over the trend that companies ban out all their in-house software development. For instance, the road model case could have accomplished much more impact without the necessity to convince the not very eager software vendors to implement this model as an extra I/O facility.

The software component technology in combination with the integrated development environments (visual programming) offer the opportunity to break this too far-reaching dependence from software vendors.

Samenvatting

1 Het probleem

De Bouw heeft een achterstand bij de inzet van informatie technologie (IT) in de ontwerp- en uitvoeringsfase van een bouwproject. Het unieke karakter van de producten (one-of-a-kind), het aan de locatie van het bouwwerk gerelateerde vervaardigingsproces, de per project geregelde organisatie van het productieproces, de complexe en geïntegreerde structuur van de producten zijn kritische factoren die de aarzelende inzet van technologische innovaties in de Bouw kunnen verklaren. De problemen zullen in de toekomst alleen maar groter worden. Hogere eisen op het gebied van veiligheid, milieuvervuiling, energiebesparing en kwaliteit van het eindproduct vallen samen met een toenemende concurrentiedruk en door de markt verlangde kortere bouw tijden. Het ontwerp- en vervaardigingsproces moet zodanig flexibel zijn dat late tot zeer late wijzigingen in het plan van eisen verwerkt kunnen worden. Een dergelijk flexibel productieproces is strijdig met de huidige manier van werken (start-from-scratch) in de Bouw.

De toepassing van informatie technologieën zoals Product Data Technology (PDT), Computer Integrated Construction (CIC), Concurrent en Co-operative Design/Engineering worden door andere industriesectoren met deels vergelijkbare problemen nagestreefd. Vergeleken met deze takken van industrie loopt de Bouw achter in het gebruik van hedendaagse informatie technologie voor de realisatie van een bouwproject. Hoewel al veel activiteiten door computers worden ondersteund is het niveau van de communicatie infrastructuur in een bouwproject onvoldoende om de echte voordelen hiervan te plukken. De huidige afspraken voor informatie-uitwisseling missen het noodzakelijke niveau om de gewenste flexibiliteit te kunnen ondersteunen. Het op een hoger niveau brengen van de communicatie infrastructuur kan de efficiëntie van het productieproces en kwaliteit van het eindproduct sterk verbeteren. Een hoger semantisch niveau van informatie-uitwisseling is hierbij de sleutelfactor om deze communicatie tussen computers onderling en tussen computers en mensen in de Bouw te verbeteren.

2 De voorgestelde oplossing

Representatie

Vanouds hanteert de Bouw de technische tekening als de primaire vastlegging (representatie) van bouwproducten. De tekening-representatie, die in principe bedoeld is voor menselijke interpretatie, is veel minder geschikt voor interpretatie door een computer.

Om het betekenisniveau, en meer algemeen de kwaliteit van de IT infrastructuur, op een hoger plan te brengen zouden representaties die tussen computers worden uitgewisseld aan de volgende eisen moeten voldoen:

- bij voorkeur een impliciete representatie, hieruit kunnen dan in de regel meer expliciete representaties worden afgeleid,
- een door een computer interpreteerbare representatie, om echte integratie van door computers ondersteunde processen mogelijk te maken, en
- een vormonafhankelijke representatie structuur, deze structuur maakt het mogelijk om een veelvoud aan vormmodellen aan hetzelfde object te koppelen en biedt tevens een kapstok voor alle andere vormonafhankelijke eigenschappen.

Een productmodel (dat is gebaseerd op een goed gedefinieerde, flexibele en semantisch rijke conceptuele structuur) is een representatievorm die in staat is om aan al deze eisen te voldoen. Een productmodel is een informatiemodel dat, in een neutrale vorm ten opzichte van een willekeurige toepassing, alle informatie kan vastleggen van een product gedurende één of meerdere levensfasen.

Productmodelstructuren

Productmodelleren is alleen haalbaar als de toegepaste modellen, zoals eerder opgemerkt, zijn gebaseerd op een goed gedefinieerde, flexibele en semantisch rijke conceptuele structuur. Mechanismen ter structurering zoals reikwijdte, niveau van specificatie, decompositie, concretisatie, modulariteit, connectiviteit en levenscyclusfase zijn noodzakelijke ingrediënten voor een dergelijke superstructuur. Anderzijds mag deze superstructuur geen onnodige beperkingen opleggen aan de modelleervrijheid van ontwikkelaars van producttypemodellen of hen dwingen om mechanismen te hanteren waarin ze niet geïnteresseerd zijn. Deze laatste constatering is tevens een belangrijke voorwaarde om een product en proces referentiemodel ingang te doen vinden in deze wereld.

Een flexibele productmodelstructuur is ook noodzakelijk om geschikte ophangpunten (slots) aan te bieden waaraan de verschillende vormmodellen die bij hetzelfde productmodel behoren kunnen worden aangehaakt. Als het product-

typemodel goed is gestructureerd heeft elk vormmodel een unieke locatie op basis van zijn positie ten opzichte van de verschillende modelleerdimensies. Modulariteit in vormmodellering biedt de mogelijkheid een compleet maar groot vormmodel (dat bij meer dan één ophangpunt is onder te brengen) te decomponeren in modules die één-op-één zijn gerelateerd aan ophangpunten.

Een voorbeeld product en proces referentiemodel wordt besproken dat een grote verzameling structureringsmechanismen kan ondersteunen. Deze structureringsmechanismen spannen (als evenzovele modelleerdimensies) het modelleerdomein op. Deze modelleerdimensies worden als onafhankelijk (orthogonaal) beschouwd, zodat entiteiten gevormd kunnen worden door het combineren van entiteiten uit elk van de vijf modelleerdimensies. Hoewel een product en proces referentiemodel als dit kan worden gebruikt voor het direct instantiëren van objecten zal de semantische inhoud als regel te algemeen zijn voor dit doel. De rechtvaardiging voor het bestaan van een dergelijk model is het gemak van integratie en samenwerking (inter-operability) van product en procesmodellen die voor een bepaalde tak van industrie worden ontwikkeld. Het idee is om een gelaagde hiërarchie van modellen (van breed en algemeen tot nauw en gedetailleerd) te ontwikkelen die allemaal zijn gebaseerd op een gemeenschappelijk product en proces referentiemodel. In een dergelijk raamwerk kan elke plek een model bevatten van een goed gedefinieerde en beperkte reikwijdte, dat als ouder-model kan dienen voor meer specifieke kind-modellen of als leveranciers-model voor andere klant-modellen. Het conformeren van ieder model aan de toegepaste structureringsmechanismen realiseert een 'plug-in' architectuur die herbruik van goede modellen aanmoedigt en ook de vervanging van minder presterende modellen.

Modelleren van complexe en unieke individuele vormen

Vormrepresentatie is vanouds het domein van geometrisch modelleren en computer graphics. In de loop van de jaren zijn de gegevensstructuren in geometrisch modelleren geëvolueerd om aan eisen te voldoen op het gebied van een optimale evaluatiesnelheid of het robuust uitvoeren van geometrische operaties. Productmodelleren stelt voor een deel andere eisen aan de toegepaste gegevensstructuren waaraan de huidige vormrepresentatiemethoden niet altijd kunnen voldoen. Belangrijke structureringseisen zijn bijvoorbeeld:

- het vermogen om de standaard modelleerstructuren van decompositie en connectiviteit te volgen,
- ondersteuning voor parametrische (herbruikbare) vormdefinitie,
- ondersteuning voor modulariteit en

- voldoende vrijheid voor het maken van ruimtelijke claims voor een bepaald onderdeel in de productstructuur.

Een nieuwe vormrepresentatie methode wordt geïntroduceerd die is ontwikkeld vanuit het perspectief van productmodelleren: de Space Deformation Tree representatie. De Space Deformation Tree representatie heeft verschillende eigenschappen die het in het bijzonder geschikt maken voor associatie met productmodellen:

- Het geometrische gedeelte (de eigenlijke space deformation tree) biedt de mogelijkheid een vormobject te definiëren als een parametrisch bibliotheek-object in een neutrale onvervormde ruimte. Als vervolgens de basis modelleerruimte van een dergelijke niet-gebonden vorm wordt verbonden met de actuele ruimte-graaf van een vormmodel zal het zich vormen volgens de dan geldende geometrie. Deze techniek kan het potentiële domein voor parametrische vormobjecten (en daarmee de mogelijkheden van herbruik van eerder ontworpen producten) aanmerkelijk doen toenemen.
- Het topologische gedeelte ondersteunt effectief non-manifold topologie, hetgeen een noodzakelijke voorwaarde is voor het implementeren van expliciete connectiviteit en referentiegeometrie representatie.
- Het vormgedeelte is goed afbeeldbaar op verschillende decompositiestrategieën (type/exemplaar).
- Een ruimte-graaf/cel-graaf combinatie kan als een submodel worden beschouwd (modulariteit) en kan als een macro-object op vorm niveau worden gemanipuleerd. Het integreren (samenvoegen) van ruimte-grafen of cel-grafen biedt een flexibele operatie om nieuwe vormobjecten samen te stellen uit een beperkte verzameling van vormprimitieven.
- Er worden veel mogelijkheden geboden om ruimtelijke claims voor bepaalde onderdelen van het productmodel vast te leggen. Deze claims kunnen elkaar uitsluiten of juist overlappen. Dezelfde faciliteiten kunnen worden gebruikt om ruimtelijke connectiviteit vast te leggen.

Integratie van productmodelleren en vormmodelleren

Een complex product kan door veel verschillende vormmodellen worden gerepresenteerd, er kan echter voor dit product maar één uniek productmodel bestaan. Het productmodel zal gewoonlijk een hiërarchische structuur bevatten: globale (algemeen geldende) informatie in de buurt van de oorsprong (wortel) en locale (gedetailleerde) informatie in de buurt van de uiteinden (bladeren). Deze hiërarchische structuur is ook heel geschikt om de verschillende vormmodellen te lokaliseren, daarbij kunnen de vormelementen worden verdeeld

over de uiteinden (en soms ook op de tussenliggende knooppunten) van de productboom. Vanouds worden dergelijke vormmodellen nauwelijks beperkingen opgelegd, wat hun onderlinge relaties reduceert tot het simpele feit dat ze verbonden zijn met hetzelfde productmodel. Deze situatie met vele onafhankelijk gedefinieerde vormmodellen is onvermijdelijk in een informatiesysteem met zogenaamde eilanden van automatisering, maar het is echter onacceptabel voor een geïntegreerd informatiesysteem

Criteria zijn nodig om twee vormmodellen te vergelijken en te beslissen of ze consistent met elkaar zijn en met hun gezamenlijke productmodelstructuur:

- *nauwkeurigheid*
Door vast te stellen dat de puntverzameling van de maximum omhullende van het model met de lagere nauwkeurigheid de puntverzameling van de maximum omhullende met de hogere nauwkeurigheid volledig moet bevatten, is een natuurlijke vorm van een globaal/gedetailleerd hiërarchie geïntroduceerd.
- *geharmoniseerde decompositie*
Een vormmodel kan ook langs de decompositieboom worden gestructureerd. Door het koppelen van de vormdecompositie aan de productstructuurdecompositie biedt het vormmodel evenveel refereerbare ingangen als er vertakkingen in de boom zijn, in plaats van slechts één ingang: de oorsprong (wortel). Als alle vormmodellen volgens dezelfde decompositie zijn verdeeld is aan een belangrijke eis voor vergelijking voldaan.
- *expliciete connectiviteit*
Expliciete connectiviteit biedt een additioneel criterium om de consistentie van vormmodellen te kunnen vergelijken.

3 Evaluatie van de praktijkvoorbeelden

Twee realistische praktijkvoorbeelden zijn gebruikt om de bruikbaarheid van de concepten die zijn besproken in de hoofdstukken 4 (Productmodelstructuren), 5 (Vormmodelleren) en 6 (Integratie van productmodelleren en vormmodelleren). De twee praktijkvoorbeelden zijn realistisch omdat ze zijn geselecteerd uit de lopende projecten die TNO uitvoert en/of heeft uitgevoerd in opdracht van Rijkswaterstaat gedurende de looptijd van dit onderzoek. Deze projecten bevatten echter geen bijzondere doelen ter toetsing van de concepten die in dit proefschrift worden behandeld. Met als gevolg dat veel concepten adequaat gedemonstreerd konden worden maar zeker niet allemaal. Met name

de ideeën uit hoofdstuk 6 (Integratie) worden helaas onvoldoende belicht door deze voorbeelden.

Wegen

Wegontwerp heeft altijd een vooraanstaande rol gespeeld op het gebied van product data technology. Toch blijkt de ontwikkeling van een informatiemodel voor wegen dat een hoger betekenisniveau ondersteunt dan een kaal minimum van expliciete geometrie (bijvoorbeeld een gelabelde verzameling van 3D strengen) geen eenvoudige zaak.

- *productmodelleren*

De wegen-case toont aan dat productmodelstructuren niet alleen toepasbaar zijn op producten die geassembleerd kunnen worden (met duidelijk te onderscheiden overgangen), maar dat ook producten met een meer monolithische structuur, zoals wegen, hiervoor geschikt zijn. Door het introduceren van hiërarchische structuren voor wegmodelleren vervalt de noodzaak om telkens opnieuw te beginnen na bijvoorbeeld een belangrijke wijziging in een alignement of dwarsprofiel. Een belangrijke toename van herbruikbaarheid van eerdere ontwerpen is het directe gevolg van deze aanpak. Door het scheiden van de unieke (one-of-a-kind) eigenschappen (lees: de wegalignementen) van de rest van het wegmodel (lees: de productstructuur), is het mogelijk om parametrische objectbibliotheken (b.v. standaard wegonderdelen als op- en afritten, splitsen of samenvoegen van rijbanen of rijstroken, standaard kruisingen, enz.) te introduceren voor wegmodellering.

De Road Shape Model Kernel (WegGeometrieModel) gebruikt slechts een beperkte verzameling van structureringsmechanismen, het is in essentie een decompositie van exemplaarobjecten. Ofschoon dit een aanvaardbaar model oplevert zijn bepaalde vragen lastig te checken, bijvoorbeeld met betrekking tot de continuïteit van twee aangrenzende wegsecties. Daarnaast moeten globale veranderingen (b.v. om de hellingen van alle taluds te veranderen) apart voor ieder exemplaarobject van een bepaald objecttype worden aangebracht.

- *vormmodelleren*

De Space Deformation Tree representatie heeft in dit project zijn waarde bewezen met de correcte vormevaluatie van de volledige Road Shape Model Kernel test-reeks. Zijn structuur integreert op een natuurlijke manier verschillende modelleerruimtes, wat de mogelijkheid biedt voor elke situatie de meest geëigende modelleerruimte te kiezen. De SDT representatie kan ook worden toegepast om aan allerlei continuïteitseisen te voldoen zowel in langs- als in dwarsrichting.

- *integratie van productmodelleren en vormmodelleren*

Omdat de Road Shape Model Kernel een volledig parametrische vormbeschrijving kent is de gegenereerde SDT representatie automatische gekoppeld aan de productmodelstructuur.

Viaducten

Een viaduct is een technische oplossing voor de kruising van een weg met één of meer andere wegen. Het is daarom voor de hand liggend een product-typemodel voor viaducten te relateren aan een algemeen wegmodel (zoals bijvoorbeeld het WegGeometrieModel). Twee projecten zijn hierbij beschouwd: het ViaDesign project van de Bouwdienst van Rijkswaterstaat en het Europese ESPRIT GEM (Generic Engineering analysis Model) project. Gezamenlijk bestrijken ze het gebied van het ondersteuningssysteem en de wegdraagfunctie van een viaduct.

- *productmodelleren*

De viaduct-case bevestigt dat parametrisch modelleren, dat nauw is geassocieerd met productmodelleren, een krachtige methode is om het ontwerp van naar schatting 80 à 90 procent van alle viaducten significant te versnellen. Tot voor kort stond de complexe en unieke vorm van elk viaductexemplaar parametrisatie met voldoende flexibiliteit niet toe.

Het praktijkvoorbeeld laat ook zien dat productmodelleren de mogelijkheid biedt de verschillende componenten van een groot bouwproject, die tot nog toe gescheiden werden behandeld, te combineren. Complete eenheden, bijvoorbeeld alle viaducten en wegsecties van een groot verkeersknooppunt, kunnen op een globaal infrastructuur niveau samengevoegd worden. Een betere integratie van de verschillende onderdelen zal het gevolg kunnen zijn.

- *vormmodelleren*

De SDT representatie biedt de mogelijkheid om het viaduct en het wegalignement gescheiden te modelleren, dat wil zeggen dat de parametrisatie van het viaduct een onvervormde (rechte) structuur vastlegt, terwijl de alignement specificaties het later in de gewenste vorm buigen.

De verschillende gekoppelde modelleerruimtes bieden bijvoorbeeld de mogelijkheid om de locatie van het ondersteuningssysteem in globale onafhankelijke coördinaten te specificeren, maar dat voor de positie van de geleiderail op het viaductdek in locale iso-parametrische coördinaten te doen.

- *integratie van productmodelleren en vormmodelleren*

Zie het overeenkomstige item bij de weg-case.

References

[Bakkeren 1997]

Bakkeren, W.J.C., *Computer-integrated Structural Engineering, Supporting the Structural Engineer's Participation in a Computer-integrated Construction Process*, dissertation Delft University of Technology, ISBN 90-9010193-4, 1997.

[Bakkeren and Willems 1993]

Bakkeren, W.J.C. and Willems, P.H., *Capturing and Structuring the Meaning of Communication in the Building and Construction Industry*, in proceedings Management of Information Technology for Construction, Singapore, eds. Mathur, K.S., Betts, M.P. and Wai Tham, K., pp. 435-451, World Scientific Publishing Co. Pte. Ltd., 1993.

[Böhms, Rousset, Miles, Leal and Helpenstein 1995]

Böhms, H.M., Rousset, M., Miles, C., Leal, D. and Helpenstein, H.J., *The GEM Modeling Methodology, An ISO 10303 STEP - Compliant Approach*, ESPRIT CIME 8894 - Generic Engineering-analysis Model - D1201, 1995.

[Braid 1978]

Braid, I.C., *On storing and changing shape information*, Computer Graphics 12, 3 (1978), 252-256.

[Bronsvoort and Jansen 1993]

Bronsvoort, W.F., Jansen, F.W., *Feature modelling and conversion - Key concepts to concurrent engineering*, Computers in Industry 21(1): 61-86, 1993.

[Callahan and Heisserman 1996]

Callahan, S. and Heisserman, J., *A Product Representation to Support Process Automation*, in 'Product Modeling for Computer Integrated Design and Manufacture', TC5/WG5.2 International Workshop on Geometric Modeling in Computer-Aided Design (285-296), 19 - 23 May 1996, Airlie, Virginia, U.S.A, ISBN 0-412-80980.

[CIMsteel 1995]

CIMsteel, *Part 4A: The Generic Logical Product Model*, in CIMsteel Integration Standards (Release One), Data Exchange Specification (Illustrative), edition 1.0, Department of Civil Engineering, University of Leeds, September 1995.

[Davis 1990]

Davis, E., *Representations of commonsense knowledge*, Morgan Kaufmann Publishers, Inc., ISBN 1-55860-033-7, 1990.

[Eastman 1981]

Eastman, C.M., *The design of assemblies*, Tech. Rep. 810197, SAE Technical Paper, 1981.

[Emmerik 1990]

Emmerik, J.G.M. van, *Interactive design of parameterized 3D models by direct manipulation*, Delft University Press, ISBN 90-6275-633-6, 1990.

[EPISTLE 1994]

EPISTLE Core Data Model for Exchange of STEP data in the Process Industries, R. Reschke, DEC, 1994.

- [Faux and Pratt 1979]
Faux, I.D. and Pratt, M.J., *Computational Geometry for Design and Manufacture*, Ellis Horwood Series in Mathematics and its applications, ISBN 0-470-27069-1, 1979.
- [Gielingh 1988]
Gielingh, W.F., *General AEC Reference Model (GARM)*, TNO Building and Construction Research, BI-88-150, October 1988.
- [Gielingh and Goult 1993]
Gielingh, W.F. and Goult, R., *Requirements for Model Integration and Interoperability*; European Workshop on Model Integration and Interoperability, Darmstadt, August 1993.
- [Gielingh and Suhm 1993]
Gielingh, W.F. and Suhm, A.K. (ed.), *IMPACT Reference Model: An Approach for Integrated Product and Process Modelling of Discrete Parts Manufacturing*, vol. 1, Research Reports ESPRIT: Project 2165 - IMPACT, Springer-Verlag, Darmstadt, Germany, 1993.
- [Gielingh, Braun, Beeckman and Willems 1994]
Gielingh, W.F., Braun, S., Beeckman, D. and Willems, P.H., *The PISA Product and Process Model*, in proceedings CAD'94, Paderborn, Germany, 1994.
- [Goranson 1992]
Goranson, H.T., *Dimensions of Enterprise Integration*; in: Proceedings of First International Conference on Enterprise Integration Modelling; MIT press, Cambridge Mass., July 1992.
- [ISO/TC184 1994a]
ISO 10303-1, *Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and Fundamental Principles*, Geneva, 1994.
- [ISO/TC184 1994b]
ISO 10303-11, *Industrial automation systems and integration - Product data representation and exchange - Part 11: EXPRESS Language Reference Manual*, Geneva, 1994.
- [ISO/TC184 1994c]
ISO 10303-41, *Industrial automation systems and integration - Product data representation and exchange - Part 41 : Integrated generic resources : Fundamentals of product description and support*, Geneva, 1994.
- [ISO/TC184 1994d]
ISO 10303-42, *Industrial automation systems and integration - Product data representation and exchange - Part 42 : Integrated generic resources : Geometric and topological representation*, Geneva, 1994.
- [Kirkley and Seitz 1991]
Kirkley, J.R., and Seitz, B.K., *STEP Framework - Concepts and Principles*, ISO/TC184/SC4/WG5/P1, July 1991.
- [Krause and Lüddemann 1996]
Krause, F.L. and Lüddemann, J., *Virtual Clay Modelling*, in 'Product Modeling for Computer Integrated Design and Manufacture', TC5/WG5.2 International Workshop on Geometric Modeling in Computer-Aided Design (162-178), 19 - 23 May 1996, Airlie, Virginia, U.S.A, ISBN 0-412-80980.
- [Kuiper 1989]
Kuiper, P., *ProMod 4.1 Reference Manual*, TNO Building and Construction Research, BI-89-137, 1989.

- [Krom 1997]
Krom, R.P., *Robots in the Building Industry* Thesis Delft University of Technology, ISBN 90-9010974-9, Sassenheim, November 1997.
- [Leal, Böhms, Cazeaux, Goult, Helpenstein and Korwaser 1997]
Leal, D., Böhms, H.M., Cazeaux, J.-P., Goult, R., Helpenstein, H. and Korwaser, H., *Generic Engineering-analysis Model, Practical Implementation Issues for GEM*, ESPRIT CIME 8894 - Generic Engineering-analysis Model - D2404, 1997.
- [Lee and Gossard 1985]
Lee, K. and Gossard, D.C., *A hierarchical datastructure for representing assemblies*, Computer Aided Design 17, 1 (1985), 15-19.
- [Luijten 1996]
Luijten, B., *PMshell User's Guide*, TNO Building and Construction Research, 1996.
- [Luiten 1994]
Luiten, G.T., *Computer Aided Design for Construction in the Building Industry*, Thesis Delft University of Technology, ISBN 90-9006779-5, The Hague, 1994.
- [Mäntylä 1988]
Mäntylä, M., *An Introduction to Solid Modeling*, Computer Science Press, Rockville, 1988.
- [Neider 1993]
Neider, J., *OpenGL programming guide: the official guide to learning OpenGL*, release 1 / OpenGL Architecture Review Board; Addison-Wesley Publishing Company, ISBN 0-201-63274-8, 1993.
- [OSA 1989]
ESPRIT Consortium AMICE, *Open System Architecture for CIM (CIM-OSA)*, Research Reports ESPRIT Project 688, Vol. 1, Brussels (Ed.), ISBN 3-540-52058-9, Springer Verlag, 1989.
- [Park and Lee 1996]
Park, S. and Lee, K., *Representation of Geometric Tolerances and its Application to Verify Assemblability between Toleranced Parts*, in 'Product Modeling for Computer Integrated Design and Manufacture', TC5/WG5.2 International Workshop on Geometric Modeling in Computer-Aided Design (297-307), 19 - 23 May 1996, Airlie, Virginia, U.S.A, ISBN 0-412-80980.
- [PISA 1996]
The ESPRIT-PISA Project Consortium, *The PISA Project: A Survey on STEP*, Shaker Verlag, Aachen, ISBN 3-8265-1118-2, 1996.
- [Rappoport 1993]
Rappoport, A.A., *Scheme for single instance representation in hierarchical assembly graphs*, in IFIP Conference on Geometric Modeling in Computer Graphics, Genova, Italy, June 1993, B. Falcidieno and T.L. Kunii, Eds. Springer 1993.
- [Requicha 1980]
Requicha, A.A.G., *Representations of Rigid Solids - Theory, Methods, and Systems*, ACM Computing Surveys, Vol. 12(4): 437-464, 1980.
- [Requicha 1983]
Requicha, A.A.G., *Toward a theory of geometric tolerance*, International Journal of Robotic Research, 2(4): 45-60, 1983.
- [Smith 1986]
Smith, B.M., *PDES Initiative Activities*, NIST, 1986.

[SofTech 1981]

SofTech, *Function Modeling Manual (IDEF0)*, vol.4, Integrated Computer-Aided Manufacturing (ICAM), Architecture Part II, SofTech, Inc., Waltham, USA, 1981.

[Sowa 1984]

Sowa, J.F., *Conceptual Structures: Information Processing in Mind and Machine*, The systems programming series, Addison-Wesley Publishing Company, Inc., 1984.

[Tolman, Böhms, Nederveen and Bakkeren 1994]

Tolman, F.P., Böhms, H.M., Nederveen, G.A.van, and Bakkeren, W.J.C., *LSE Project Type Model, version 1.0*, ESPRIT 7280 - ATLAS, final, public, D106-I, November 1994.

[VRML 1996]

ISO/IEC/JTC1/SC24, Information Technology, Computer Graphics and Image Processing, and the VRML Architecture Group (VAG), *The Virtual Reality Modeling Language Specification, Version 2.0*, ISO/IEC CD 14772, August 1996,
<http://vag.vrml.org/VRML2.0/FINAL/spec/>

[Weiler 1986]

Weiler, K., *Topological Structures for Geometric Modeling*, PhD. Thesis, Rensselaer Polytechnic Institute, 1986.

[Wesley 1980]

Wesley, M.A., Lozano-Perez, T., Lieberman, L.I., Lavin, M.A. and Grossman, D.D., *A geometric modeling system for automated assembly*, IBM J. Res. Develop. 24, 1 (Jan. 1980), 64-74.

[Willems 1988]

Willems, P.H., *A Meta-Topology for Product Modelling*, in proceedings 'Conceptual Modelling of Buildings', in Lund, Sweden, ed. Christiansson, Per and Karlsson, Henry, pp. 213-221, Swedish Building Centre, 1988 (CIB W78 and W74).

[Willems 1989]

Willems, P.H., *Meta-topologie en produktmodelleren*, (in Dutch), in proceedings CAPE Nederland '89, 9-11 May 1989, Amsterdam.

[Willems 1990a]

Willems, P.H., *Road Model Kernel*, TNO Building and Construction Research, B-89-831, March 1990.

[Willems 1990b]

Willems, P.H., *Standaardisatie, Eindstation of startpunt voor onderzoek?*, (in Dutch), De Bouwadviseur, December 1990, pp. 36-38.

[Willems 1991]

Willems, P.H., *STEP in de Bouw*, article in 'Handboek CAD/CAM deel II', Samsom BedrijfsInformatie, October 1991.

[Willems 1993]

Willems, P.H., *Semantic Topology: the Management of Shape Definition*, in proceedings 'The Management of Information Technology for Construction', First International Conference, 17 - 20 August 1993, Singapore.

[Willems 1995]

Willems, P.H., *Documentatie WegGeometrie Model*, (partly in Dutch), TNO Building and Construction Research, 95-BI-R1589, 1995.

[Willems 1996]

Willems, P.H., *Modelling Structure and Shape in the Construction Industry*, in 'Product Modeling for Computer Integrated Design and Manufacture', TC5/WG5.2 International

Workshop on Geometric Modeling in Computer-Aided Design (94-103), 19 - 23 May 1996, Airlie, Virginia, U.S.A, ISBN 0-412-80980.

[Wix, Tolman, Poyet and Björk 1994]

Wix, J., Tolman, F.P., Poyet, P. and Björk, B.C., *Part 106: Building Construction Core Model*, ISO TC184/SC4/WG3 N341, 1994.

Road Shape Model Kernel

SCHEMA Chain_UoF;

(*

Chain

A chain is defined to contain a set of elements (ChainElement) that form together a contiguous one-dimensional chain of non-overlapping links. The chain has a distinct first element (no predecessor) and last element (no successor), hence circular chains cannot be composed.

EXPRESS specification:

*)

```
ENTITY Chain
  elements                : LIST [1:?] OF ChainElement;
DERIVE
  firstElement            : ChainElement := body
  lastElement             : ChainElement := body
END_ENTITY;
```

(*

Attribute definitions:

elements

The elements form the list of all participating unique elements.

firstElement

The first element has no predecesing element.

lastElement

The last element has no succeeding element.

ChainElement

A chain element is defined to be a link in a chain. The first element of a chain has no predecessor while the last element has no successor.

EXPRESS specification:

*)

```
ENTITY ChainElement
DERIVE
  chain                  : Chain := body
  predecessor            : ChainElement := body
```

```

    successor                : ChainElement := body
END_ENTITY;

```

```
( *
```

Attribute definitions:

chain

The chain refers to the chain this element is participating in.

predecessor

The predecessor refers to the preceding element (except for the first element).

successor

The successor refers to the succeeding element (except for the last element).

LongChain

A long(itudinal) chain is defined to contain a set of longitudinal elements (LongChainElement) that form together a contiguous one-dimensional chain of non-overlapping links. The chain is aligned along a longitudinal axis that need not to be a straight line but, more relaxed, may be a continuous curve.

EXPRESS specification:

```
* )
```

```

ENTITY LongChain
  SUBTYPE OF (Chain);
  SELF\Chain.elements : LIST [1:?] OF LongChainElement;
  startLongChainage    : REAL;
DERIVE
  totalLongSize        : REAL := body
  longAnchor           : LongChainElement := body
END_ENTITY;

```

```
( *
```

Attribute definitions:

elements

startLongChainage

The start long(itudinal) chainage is the longitudinal co-ordinate of the start of the first element.

totalLongSize

The total long(itudinal) size is the sum of all the long(itudinal) sizes of the participating elements.

longAnchor

The long(itudinal) anchor element is either the element that embeds the origin of the longitudinal axis or is the element that is nearest to this origin.

LongChainElement

A long(itudinal) chain element is defined to be a link in a long(itudinal) chain.

EXPRESS specification:

```
* )

ENTITY LongChainElement
  SUBTYPE OF (ChainElement);
  longSize          : REAL;
DERIVE
  longOffset        : REAL := body
WHERE
  INV_longSize:
END_ENTITY;

( *
```

Attribute definitions:

longSize
The long(itudinal) size is the primary dimension of the element measures along the longitudinal direction.

longOffset
The long(itudinal) offset calculates the offset of the start of the element to the origin of the longitudinal axis.

Formal propositions:

INV_longSize
The element long(itudinal) size must be greater than zero.

TransChain

A trans(versal) chain is defined to contain a set of transversal elements (TransChainElement) that form together a contiguous one-dimensional chain of non-overlapping links. The chain is aligned along a transversal axis.

EXPRESS specification:

```
* )

ENTITY TransChain
  SUBTYPE OF (Chain);
  SELF.Chain.elements : LIST [1:?] OF TransChainElement;
  startTransChainage  : REAL;
DERIVE
  transAnchor         : TransChainElement := body
  totalTransSize      : REAL := body
END_ENTITY;

( *
```

Attribute definitions:

elements

startTransChainage

The start trans(versal) chainage is the transversal co-ordinate of the start of the first element.

transAnchor

The trans(versal) anchor element is either the element that embeds the origin of the transversal axis or is the element that is nearest to this origin.

totalTransSize

The total trans(versal) size is the sum of all the trans(versal) sizes of the participating elements.

TransChainElement

A trans(versal) chain element is defined to be a link in a trans(versal) chain.

EXPRESS specification:

*)

ENTITY TransChainElement

SUBTYPE OF (ChainElement);

transSize : REAL;

DERIVE

transOffset : REAL := body

END_ENTITY;

(*

Attribute definitions:

transSize

The trans(versal) size is the secondary dimension of the element measures along the transversal direction.

transOffset

The trans(versal) offset calculates the offset of the start of the element to the origin of the transversal axis.

*)

END_SCHEMA;

SCHEMA HVAlignment_UoF;

(*

HVElement

The HVElement entity is an abstract supertype entity that is able to attach a linear curvature function to its definition interval on the longitudinal axis. This curvature function is defined by specifying the curvatures at the start and the end of the interval.

EXPRESS specification:

```
* )

ENTITY HVElement
  SUBTYPE OF (LongChainElement);
  startCurvature      : REAL;
  endCurvature        : REAL;
WHERE
  INV_curve:
END_ENTITY;

( *
```

Attribute definitions:

startCurvature
 The startCurvature is the curvature at the start of the HVElement.

endCurvature
 The endCurvature is the curvature at the end of the HVElement.

StraightElement

A straight element is a HVElement with a constant curvature of zero. A straight element may either participate in a horizontal alignment as well as in a vertical alignment.

EXPRESS specification:

```
* )

ENTITY StraightElement
  SUBTYPE OF (HVElement);
WHERE
  INV_curve:
END_ENTITY;

( *
```

Formal propositions:

INV_curve
 Both curvature attributes should be zero valued.

CircularElement

A circular element is a HVElement with a constant non-zero curvature. A circular element may either participate in a horizontal alignment as well as in a vertical alignment.

EXPRESS specification:

```

*)

ENTITY CircularElement
  SUBTYPE OF (HVElement);
DERIVE
  curvature          : REAL := body
  radius             : REAL := body
  angle              : REAL := body
WHERE
  INV_curve:
END_ENTITY;

( *

```

Attribute definitions:

curvature
The curvature attribute holds the constant non-zero curvature value.

radius
The radius attribute holds the reciproke curvature value (if curvature $\neq 0.0$).

angle
The angle attribute holds the difference angle between the end direction and the start direction.

Formal propositions:

INV_curve
Both curvature attributes are equal but not zero valued.

ClothoidElement

A clothoid element is a HVElement with a linear non-constant curvature function. A clothoid element may only participate in a horizontal alignment.

EXPRESS specification:

```

*)

ENTITY ClothoidElement
  SUBTYPE OF (HVElement);
DERIVE
  startRadius        : REAL := body
  endRadius          : REAL := body

  angle              : REAL := body
  a2                 : REAL := body
  origin             : REAL := body
WHERE
  INV_curve:
END_ENTITY;

( *

```

Attribute definitions:

startRadius

The startRadius attribute holds the reciproke startCurvature value (if startCurvature \neq 0.0).

endRadius

The endRadius attribute holds the reciproke endCurvature value (if endCurvature \neq 0.0).

angle

The angle attribute holds the difference angle between the end direction and the start direction.

a2

The a2 attribute holds the clothoid constant.

origin

The origin attribute calculates the distance to the clothoid origin.

Formal propositions:

INV_curve

The startCurvature is not equal to the endCurvature.

HorizontalAlignment

A horizontal alignment defines the horizontal geometry of the road axis projected on the xy-plane. The horizontal alignment is assembled from a set of elements of three different types: StraightElement, CircularElement or ClothoidElement.

EXPRESS specification:

*)

```
ENTITY HorizontalAlignment
  SUBTYPE OF (LongChain);
  anchorX          : REAL;
  anchorY          : REAL;
  anchorAngle      : REAL;
  SELF\LongChain.elements : LIST [1:?] OF HVElement;
WHERE
  INV_angle:
END_ENTITY;
```

(*

Attribute definitions:

anchorX

The anchorX attribute holds the x co-ordinate at the origin of the longitudinal axis if the alignment includes this origin. If not it is either the x co-ordinate at the start of the alignment (startChainage \geq 0.0) or at the end of the alignment (startLongChainage + totalLongSize \leq 0.0).

anchorY

The anchorY attribute holds the y co-ordinate at the origin of the longitudinal axis if the alignment includes this origin. If not it is either the y co-ordinate at the start of the alignment ($\text{startLongChainage} \geq 0.0$) or at the end of the alignment ($\text{startLongChainage} + \text{totalLongSize} \leq 0.0$).

anchorAngle

The anchorAngle attribute holds the angle ($\text{grad} = 2 \cdot \text{PI} / 400$) at the origin of the longitudinal axis if the alignment includes this origin. If not it is either the angle at the start of the alignment ($\text{startLongChainage} \geq 0.0$) or at the end of the alignment ($\text{startLongChainage} + \text{totalLongSize} \leq 0.0$).

elements*Formal propositions:***INV_angle**

The value domain of anchorAngle should lie between -400.0 and +400.0.

VerticalAlignment

A vertical alignment defines the vertical geometry of the road axis projected on the sz-plane, where the s-axis is the curve defined by the horizontal alignment. The vertical alignment is assembled from a set of elements of two different types:

StraightElement or CircularElement.

EXPRESS specification:

*)

```
ENTITY VerticalAlignment
  SUBTYPE OF (LongChain);
  SELF\LongChain.elements : LIST [1:?] OF HVElement;
  anchorInclination       : REAL;
  anchorZ                 : REAL;
END_ENTITY;
```

(*

*Attribute definitions:***elements****anchorInclination**

The anchorInclination is the longitudinal gradient at the origin of the longitudinal axis if the alignment includes this origin. If not it is either the gradient at the start of the alignment ($\text{startLongChainage} \geq 0.0$) or at the end of the alignment ($\text{startLongChainage} + \text{totalLongSize} \leq 0.0$).

anchorZ

The anchorZ is the z co-ordinate at the origin of the longitudinal axis if the alignment includes this origin. If not it is either the angle at the start of the alignment ($\text{startLongChainage} \geq 0.0$) or at the end of the alignment ($\text{startLongChainage} + \text{totalLongSize} \leq 0.0$).

EXPRESS specification:

*)

END_SCHEMA;

SCHEMA Road_UoF;

(*

Road

The Road entity is a root entity within the Road Shape Model Kernel, i.e. no other entities refer to this entity. However, the same data model may hold more than one road instance. This property can be used to combine several roads in one and the same model.

A Road instance refers to the horizontal alignment, i.e. all instances that participate in a road instance must follow this alignment. The neutral line of the horizontal alignment, the line that keeps its original length, coincides with the longitudinal axis.

A Road instance is decomposed in longitudinal direction in one or more RoadSection instances.

EXPRESS specification:

*)

```
ENTITY Road
  SUBTYPE OF (LongChain);
  SELF\LongChain.elements : LIST [1:?] OF RoadSection;
  horizontalAlignment : HorizontalAlignment;
WHERE
  INV_start:
  INV_end:
END_ENTITY;
```

(*

Attribute definitions:

elements

horizontalAlignment

The horizontalAlignment attribute holds a reference to a horizontal alignment.

Formal propositions:

INV_start

The start of the road (startLongChainage) must be located in the interval that is defined by the horizontal alignment.

INV_end

The end of the road (startLongChainage + totalLongSize) must be located in the interval that is defined by the horizontal alignment.

RoadSection

The RoadSection entity is both a chain element and a chain. It participates as a longitudinal element in a Road, while it aggregates transversal elements (of type RoadElement) in a cross sectional way. The derived attribute longOffset refers to the longitudinal location in the Road, however, the startTransChainage positions the cross section in the transversal direction. The explicit attribute longSize holds the length of this RoadSection while the derived attribute totalTransSize computes in fact the total width of the RoadSection.

EXPRESS specification:

*)

```
ENTITY RoadSection
  SUBTYPE OF (LongChainElement, TransChain);
  SELF\TransChain.elements : LIST [1:?] OF RoadElement;
END_ENTITY;
```

(*

Attribute definitions:

elements

RoadElement

A RoadElement is an abstract supertype entity, i.e. it should not be instanced directly only through its descendants. RoadElements build up the transversal structure of the road (as RoadSections build up the longitudinal structure). Two types of descendants are available: the RoadElementComplex entity and the RoadElementPrimitive entity. Each RoadElementComplex may refer to its private vertical alignment. To bridge the height differences a RoadElementPrimitive must be placed between two RoadElementComplexes, either with fixed gradient or fixed width definition. As a result the RoadElements that shape the transversal definition of a RoadSection should alternate between these descendant types.

EXPRESS specification:

*)

```
ENTITY RoadElement
  SUBTYPE OF (TransChainElement);
END_ENTITY;
```

(*

RoadElementComplex

A RoadElementComplex aggregates a set of RoadElementPrimitives that share a vertical alignment. Like RoadSection the RoadElementComplex is both a chain and a chain element (or more precise: RoadElement). There is also an important difference:

in a RoadElementComplex both aspects are defined in the transversal direction. As a result the explicit attribute transSize and the derived attribute totalTransSize both refer to the same transversal dimension of a RoadElementComplex.

The explicit attribute startTransChainage indirectly positions the anchor line of the vertical alignment (chainage = 0.0).

EXPRESS specification:

```
*)  
  
ENTITY RoadElementComplex  
  SUBTYPE OF (RoadElement,  
    TransChain);  
  SELF\TransChain.elements : LIST [1:?] OF  
    RoadElementPrimitive;  
  verticalAlignment : VerticalAlignment;  
DERIVE  
  veralAnchor : RoadElementPrimitive := body  
  transAnchor : TransChainElement := body  
  totalTransSize : REAL := body  
WHERE  
  INV_start:  
  INV_end:  
  INV_left:  
  INV_right:  
  INV_anchor:  
END_ENTITY;  
  
(*
```

Attribute definitions:

```
elements  
verticalAlignment  
  The verticalAlignment attribute holds a reference to a vertical alignment.  
veralAnchor  
transAnchor  
totalTransSize
```

Formal propositions:

```
INV_start  
  The start of the RoadSection (chain.longOffset) must be located in the interval  
  that is defined by the vertical alignment.  
INV_end  
  The end of the RoadSection (chain.longOffset + chain.longSize) must be  
  located in the interval that is defined by the vertical alignment.  
INV_left  
  The type of the left adjacent RoadElement must be RoadElementPrimitive.  
INV_right  
  The type of the right adjacent RoadElement must be RoadElementPrimitive.
```

INV_anchor

The anchor point of the vertical alignment must be located in the transversal definition interval of this RoadElementComplex.

RoadElementPrimitive

A RoadElementPrimitive is an abstract entity that defines an elementary part of the road. The behaviour of a RoadElementPrimitive depends from the type of alignment from which it is an element. If the alignment type is RoadElementComplex both gradient and width are (and should be) defined independent from its neighbours. If the alignment type is RoadSection it acts as an intermediate element and either the gradient definition or the width definition is dependent from the adjacent RoadElementComplex instances. In this last case either the transSize (width) attribute or (exclusive) the gradient attribute equals to zero.

If the width definition or gradient definition is not a fixed value these variations can be defined by referring respectively to a width alignment or a superelevation alignment.

EXPRESS specification:

*)

```
ENTITY RoadElementPrimitive
  SUBTYPE OF (RoadElement);
  gradient          : REAL;
  widthAlignment    : WidthAlignment;
  superElevationAlignment: SuperElevationAlignment;
DERIVE
  horalOffset       : REAL := body
  veralOffset       : REAL := body
WHERE
  INV_inter:
END_ENTITY;

( *
```

Attribute definitions:

gradient

The gradient attribute holds the transversal gradient value.

widthAlignment

The widthAlignment attribute refers to a width alignment in case the width definition is not a constant value for this element.

superElevationAlignment

The superElevationAlignment attribute refers to a super elevation alignment in case the transversal gradient definition is not a constant value for this element.

horalOffset

veralOffset

Formal propositions:

INV_inter

The size and gradient attributes can never both be zero.

CarriageWay

A CarriageWay is an instantiable RoadElementPrimitive entity. For the moment no additional attributes or Where rules are defined for this entity.

EXPRESS specification:

*)

```
ENTITY CarriageWay
  SUBTYPE OF (RoadElementPrimitive);
END_ENTITY;
```

(*

Verge

A Verge is an instantiable RoadElementPrimitive entity. For the moment no additional attributes or where rules are defined for this entity.

EXPRESS specification:

*)

```
ENTITY Verge
  SUBTYPE OF (RoadElementPrimitive);
END_ENTITY;
```

(*

Slope

A Slope is an instantiable RoadElementPrimitive entity. For the moment no additional attributes or where rules are defined for this entity.

EXPRESS specification:

*)

```
ENTITY Slope
  SUBTYPE OF (RoadElementPrimitive);
END_ENTITY;
```

(*

EXPRESS specification:

```
* )

END_SCHEMA;

SCHEMA SuperElevationAlignment_UoF;

( *
```

SuperElevationAlignment

A super elevation alignment defines the axial rotation geometry for a RoadElement-Primitive. The super elevation alignment is assembled from a set of elements of two different types: ConstantSuperElevationElement or ChangingSuperElevationElement.

EXPRESS specification:

```
* )

ENTITY SuperElevationAlignment
  SUBTYPE OF (LongChain);
END_ENTITY;

( *
```

SuperElevationElement

The SuperElevationElement entity is an abstract supertype entity that is able to attach a linear transversal gradient function to its definition interval on the longitudinal axis. This transversal gradient function is defined by specifying a gradient increase over the interval.

EXPRESS specification:

```
* )

ENTITY SuperElevationElement
  SUBTYPE OF (LongChainElement);
  superElevationIncrease: REAL;
  rotationAxis          : REAL;
WHERE
  INV_incr:
END_ENTITY;

( *
```

Attribute definitions:

superElevationIncrease

The superElevationIncrease attribute holds the increase of super elevation over the super elevation element interval definition.

rotationAxis

The rotationAxis attribute holds the offset of the rotation axis with regard to the longitudinal axis.

ConstantSuperElevationElement

A ConstantSuperElevationElement is an instantiable SuperElevationElement with a superElevationIncrease of zero.

EXPRESS specification:

*)

```
ENTITY ConstantSuperElevationElement
  SUBTYPE OF (SuperElevationElement);
WHERE
  INV_incr:
END_ENTITY;
```

(*

Formal propositions:

INV_incr

The superElevationIncrease of a ConstantSuperElevationElement can only be zero.

ChangingSuperElevationElement

A ChangingSuperElevationElement is an instantiable SuperElevationElement with a superElevationIncrease that differs from zero.

EXPRESS specification:

*)

```
ENTITY ChangingSuperElevationElement
  SUBTYPE OF (SuperElevationElement);
WHERE
  INV_incr:
END_ENTITY;
```

(*

Formal propositions:

INV_incr

The superElevationIncrease of a ChangingSuperElevationElement can only differ from zero.

EXPRESS specification:

```
*)

END_SCHEMA;

SCHEMA WidthAlignment_UoF;

( *
```

WidthAlignment

A width alignment defines the transversal geometry for a RoadElementPrimitive. The width alignment is assembled from a set of elements of two different types: ConstantWidthElement or ChangingWidthElement.

EXPRESS specification:

```
*)

ENTITY WidthAlignment
  SUBTYPE OF (LongChain);
  SELF\LongChain.elements : LIST [1:?] OF WidthElement;
END_ENTITY;

( *
```

Attribute definitions:

elements

WidthElement

The WidthElement entity is an abstract supertype entity that is able to attach a linear transversal dimension function to its definition interval on the longitudinal axis. This transversal dimension function is defined by specifying a width increase over the interval.

EXPRESS specification:

```
*)

ENTITY WidthElement
  SUBTYPE OF (LongChainElement);
  startWidthIncrease : REAL;
  endWidthIncrease   : REAL;
  circular            : BOOLEAN;
DERIVE
  tangent              : REAL := body
WHERE
  INV_incr:
END_ENTITY;
```

(*

Attribute definitions:

startWidthIncrease

The startWidthIncrease attribute holds the increase of the transversal dimension at the start of the width element with regard to the start width of the referencing road element primitive.

endWidthIncrease

The endWidthIncrease attribute holds the increase of the transversal dimension at the end of the width element with regard to the start width of the referencing road element primitive.

circular

The circular attribute is a boolean switch that specifies the contour curve to be smooth (true) or linear (false).

tangent

The tangent attribute calculates the tangent of the width element.

ConstantWidthElement

A ConstantWidthElement is an instantiable WidthElement with a startWidthIncrease that equals the endWidthIncrease.

EXPRESS specification:

*)

```
ENTITY ConstantWidthElement
  SUBTYPE OF (WidthElement);
WHERE
  INV_incr:
END_ENTITY;
```

(*

Formal propositions:

INV_incr

The width increase over the interval definition of the element is constant.

ChangingWidthElement

A ChangingWidthElement is an instantiable WidthElement with a startWidthIncrease that differs from the endWidthIncrease.

EXPRESS specification:

*)

```
ENTITY ChangingWidthElement
  SUBTYPE OF (WidthElement);
```

```
WHERE  
  INV_incr:  
END_ENTITY;
```

```
( *
```

Formal propositions:

```
INV_incr
```

The width increase over the interval definition of the element is not constant.

EXPRESS specification:

```
* )
```

```
END_SCHEMA;
```

Curriculum Vitae

Family name: Willems
First names: Peter Henri
Date of birth: March 1, 1953
Place of birth: Den Haag
Civil status: Married, two children

1965-1973	Atheneum B, St.-Janscollege, Den Haag
1973-1974	Military service
1974-1986	MSc degree in Civil Engineering Delft University of Technology Majored in numerical mechanics Supervisor: Prof. dr. ir. J. Blaauwendraad Thesis: The initial design and implementation of the discrete element method package TILLY
1976-1977	Member of the board of the students' association "Practische Studie"
1977-1978	Member of the board of the faculty of Civil Engineering
1978-1979	Member of the council of the faculty of Civil Engineering
1979-1984	Educational assistant for computer applications for civil engineering
1984-1985	Staff member at the section for fluid mechanics.
1985-present	Research scientist at TNO Building and Construction Research
1990	TNO Bouw award the "Ligtenberg Penning"
1991-1998	PhD research, Delft University of Technology, Faculty of Civil Engineering

Index

3

3D model, 23, 24, 25, 28, 103, 142, 147, 181

A

absolute addressing, 52
absolute occurrence, 40, 42, 97
accuracy value, 106, 108, 109, 116, 122
accuracy zone, 105, 106, 109
activity modelling, 35
aggregation level, 34
alignment
 horizontal, 127, 130, 131, 145, 159
 super elevation, 129, 137, 143
 vertical, 127, 130, 133, 135, 143, 144, 151, 159
 width, 129, 137, 138, 143
anchor angle, 132
anchor location, 130
anchor point, 17, 132, 138, 143, 144, 145, 197
assembly object, 44
assembly/part structure, 41
ATLAS, 33
axiomatic shape primitive set, 92

B

BCCM, 33
bill of materials, 26, 42
Boolean operation, 83
boundary cell, 81, 83
boundary point, 78, 80, 81
bounded_by cell, 80
bounding cell, 80

C

cardinality, 82, 155
cardinality constraint, 49, 60, 61
cell forest, 89
cell integration, 89
cell modelling, 77
CIM-OSA, 33
CIMsteel, 33
closed boundary, 81
COMBINE, 32
communication
 human-to-human, 13, 25
 machine-to-machine, 25
concept definition, 48, 53
concept object, 50, 51, 121
conceptual model, 14, 25, 32, 33, 34, 36, 44
concretisation, 45, 46, 47, 51
concurrent engineering, 35, 44, 116

connectivity, 35, 47, 48, 115
connectivity interface, 51
connectivity network, 48, 49, 50, 51
constraint-based modelling, 47
construction phase, 55
Constructive Solid Geometry, 17, 47
container object, 47, 48
co-operative design, 24
co-ordinate system, 17, 43, 68, 69, 113, 148
co-ordinates tuple, 68
cuboid extent, 105
curvature function, 131
curvature integration, 131

D

data
 compression, 15
 product definition, 22, 26
decomposition, 35, 41, 47, 49, 52, 53, 113
 alignment, 127
 concept, 121
 harmonised, 122
 implicit, 49
 longitudinal, 133
 mixed type/occurrence, 43, 46, 49, 52, 113
 modular, 32, 45, 53, 113
 modular occurrence, 46
 modular type, 46
 non-modular occurrence, 47
 occurrence, 42, 43, 47, 49, 113
 port, 121
 property, 113
 shape, 113, 122
 transversal, 134
 type, 42, 43, 47, 49, 113
decomposition layer, 50
decomposition mechanism, 49
decomposition structure, 41, 43, 44, 45, 46
decomposition tree, 41
decompositions
 transversal, 133
definition
 shape, 11, 23, 104, 163
deformation component, 70, 71, 72, 98, 141, 146
deformation zone, 146
derived shape primitive set, 92, 93
design stage, 10, 25, 41, 103, 104, 168
destruction phase, 55
digital terrain model, 128
dimension, 58
 concretisation, 59, 110
 connectivity, 58
 modelling, 63
 specification, 61, 112

dimensionality, 66, 69, 75, 77, 79, 80, 92, 107, 108, 119
drawing, 25, 27

E

embedded space, 69, 70, 72
embedding space, 69, 70, 72, 75
enclosed cell, 79
enclosing cell, 79
energy calculation, 32
EPISTLE, 33
exclusive aggregation, 95
EXPRESS, 125
extension, 15
external connectivity, 54

F

feature modelling, 26
functional unit, 135

G

GARM, 32, 46, 55, 125
GEM, 33, 149, 155, 159
geometric constraint, 114
geometric modelling, 65
geometrical connection, 83
graph
 cell, 79, 89, 91, 100, 141
 decomposition, 45
 directed a-cyclic, 42, 72, 77, 89, 91, 113
 road shape, 145
 shape, 91, 94, 96, 97, 141, 142
 space, 72, 74, 77, 78, 82, 89, 91, 95, 100, 141
 tree, 42, 44
grouping mechanism, 47

H

Hamburger diagram, 46
hierarchy
 parent/child, 39
human interpretation, 21, 22, 26, 27, 162

I

IAI, 33
IDM, 32
IFC, 33
IMPACT, 32, 33
inclusive aggregation, 94
information definition language, 125
information modelling language, 40, 58
information object, 38, 41, 47
intension, 15
interface object, 51
interior point, 78
interior zone, 106
internal connectivity, 54
interoperability, 101

K

knowledge
 common-sense, 14

L

language
 conceptual, 15
layered structure, 45
leaf cell, 89
level
 generic, 39
 occurrence, 39
 specification, 32, 38, 40, 62, 162
 type, 39
level of detail, 45, 104
life cycle, 26, 28, 33, 34, 37, 38, 41, 55, 56, 62, 63, 125, 162, 168
location component, 69
logical point set, 85
longitudinal location, 137
longitudinal size, 133, 142
LSE, 33

M

machine interpretation, 26, 27, 162
manufacturing stage, 10, 25, 168
mapping function, 70, 92
MaristEP, 32
MARITIME, 33
meaning triangle, 15, 16
modelling cell, 77, 79
modelling mechanism, 36, 45, 46, 49, 55, 57, 58
modelling space, 66, 69, 147
modular connectivity, 51, 53, 116
modular shape model, 47
modular structure, 44
modularity, 44, 45, 49, 53, 63, 101
monolithic structure, 127
MOSS, 126

N

NIDDESC, 32
non-manifold topology, 55, 100, 164

O

occurrence object, 40, 42, 43, 44, 46, 48, 110, 112, 147
operation phase, 55
orientation component, 69
orientation vector, 69, 105, 117
orthographic projection, 22

P

PDES, 31
peripheral concept, 37
physical model, 23
physical point set, 85
PIARC, 125
PISA, 33, 35, 57

PMshell, 3, 147
point set, 66, 70, 73, 85, 95, 110, 122
port
 external, 52
 free, 121
 internal, 52
port object, 51, 53, 116, 119, 121
port occurrence, 53
port type, 53
POSC, 34
primary orientation, 70, 118, 119
primary orientation vector, 117
process modelling, 11, 35, 36, 86
product data structure, 34, 35, 63
product data technology, 34, 35, 161, 165, 170
product definition unit, 84, 85, 87, 91
product model, 25, 26, 28, 29, 31, 35, 37, 40, 55, 60, 62,
 63, 64, 65, 100, 101, 104, 111, 117, 122, 125, 149,
 162, 163, 164, 165, 166, 167, 168, 169
product model structure, 34, 104, 122, 148
product modelling, 28, 31, 91, 103, 123, 147, 160, 167,
 168, 169
product type model, 32, 60, 63, 125
ProMod, 55
property definition, 48, 53
proposed object, 45, 110

R

real world object, 40, 42, 104
redundancy, 43, 130
reference geometry, 32, 54, 55, 100, 111, 164
reference model, 35, 45, 46, 54, 58, 59, 60, 63, 163
referent, 15
relation
 adjacency, 19
 boundary, 80, 84
 connectivity, 48, 51, 53
 decomposition, 42, 43
 embedded_by, 72, 76
 embedding, 72, 75
 enclosure, 79, 84
 value inheritance, 39
relative addressing, 53
relative occurrence, 43, 52, 97
representation, 15
 boundary, 19, 160
 explicit, 16, 19, 27, 128, 162
 geometry/topology, 18
 implicit, 16, 17, 27, 28, 162
 machine-interpretable, 27, 162
 parametric, 17, 128
 pixel, 21
 shape, 100, 101, 103, 104, 126
 solid model, 87
 sweep, 17, 18
 vector, 19, 20, 21, 27
required object, 45, 110
ring of Moebius, 96
road model, 128, 147, 149
Road Model Kernel, 32, 125, 126
road modelling, 147
Road Shape Model Kernel, 125, 126, 128, 147, 149, 150,
 155

root cell, 89
root space, 72, 73, 92, 96, 100
rotation, 47, 146, 159

S

scaling, 47, 142
secondary orientation, 70
secondary orientation vector, 117, 118
set operation, 26, 47
shape
 complex, 17, 18, 91, 94, 95, 96, 99, 147
 generic, 112
 idealised, 108
 maximum, 107, 122
 minimal, 107
 nominal, 106, 107
 occurrence, 97, 112
 parametric, 100, 112
 port, 115, 120
 primitive, 17, 91, 92, 95, 113
 profile, 18
 proposed, 110, 114
 realised, 110
 required, 106, 110, 114
 result, 18, 99
 specific, 19, 97, 112
shape evaluation, 140, 147, 148, 154, 158
shape model, 63, 64, 84, 100, 103, 104, 111, 117, 122,
 123
shape modelling, 40, 63, 64, 65, 91, 100, 103, 123, 148,
 151, 160
shape type, 97
solid modelling, 23, 91
Space Deformation Tree representation, 65, 96, 100, 101,
 140, 148, 154, 158
space integration, 74, 89
space modelling, 66, 77
spatial constraint, 38, 110
spatial enumeration, 21
spatial envelope, 110, 114
spatial integrity, 146
specification
 generic, 40
 occurrence, 38, 40
 type, 40
spherical extent, 105
STEP, 31, 125, 147, 155, 160
STEP Framework, 34
stress calculation, 104
structuring mechanism, 29, 32, 34, 35, 48, 49, 53, 57, 58,
 63, 147, 163, 166, 168
 horizontal, 47
 vertical, 47
surface modelling, 23, 142
sweep
 rotational, 18
 translation, 18

T

technical solution, 1, 45, 125, 150, 166
tolerance factor, 105
tolerance modelling, 47

topological connection, 83
topological constraint, 114, 115
translation, 47, 69, 77, 141, 146
transversal gradient, 135, 138, 144, 159
transversal position, 134, 135
transversal size, 135, 138
type object, 43, 44, 46, 110

U

universe of discourse, 11, 32, 53

V

value inheritance, 39, 112
ViaDesign, 149, 150
void, 85
VRML, 47, 104, 142, 159

W

WEGEN, 126