A Meta-Topology for Product Modeling.[1]

Peter Willems
TNO-IBBC
P.O.Box 49, 2600 AA Delft
The Netherlands

Abstract

A major issue in product modeling is the integration of two essentially different modeling approaches: the top-down functional-oriented approach, and the bottom-up technical-oriented approach.
The ISO-STEP General AEC Reference Model (GARM) supports this dual design principle around the kernel entities Functional Unit and Technical Solution.
During the development of GARM a number of topology related issues were encountered. To mention two main issues:

- How to structure a functional network to be consistent over several decomposition/aggregation levels, as well as over the branches of the hierarchical tree?
- How to relate this network and the multiple coexisting representations which share this same kernel?

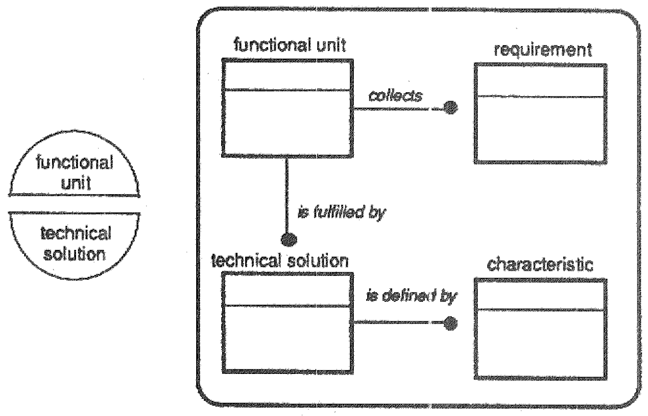
Both issues can be addressed, in principle, using a pure topology independent network and an intermediate layer to relate the dependent representations.
This intermediate layer is called: *Meta-Topology*.

Introduction

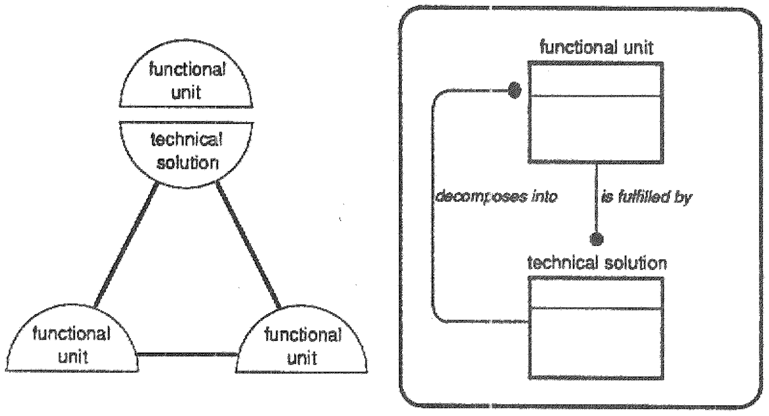
For several years product modeling is a major research topic at the TNO Institute for Building Materials and Structures (TNO-IBBC). One goal of this effort is to participate actively in the development of the ISO standard for the exchange of product model data (STEP). As a result we developed a reference model for AEC product model data which was accepted by ISO TC184/SC4/WG1 as the General AEC Reference Model [1].
The core of this reference model is based on two fundamental views to look at a product:
- the functional view, represented by the entity class *Functional Unit*, and
- the technical view, represented by the entity class *Technical Solution*.

The Functional Unit part collects all requirements and constraints which must be fulfilled. The Technical Solution part describes a possible solution to meet those requirements. Generally more than one Technical Solution can be found for the same Functional Unit. Evaluation of those solutions should reveal which Technical Solution fits best the specified functional behavior. The figures in this paper represent this duality by two matching crescent-shaped symbols, as shown in the next figure.

[1]Paper for CIB meeting, Lund, Sweden, October 1988



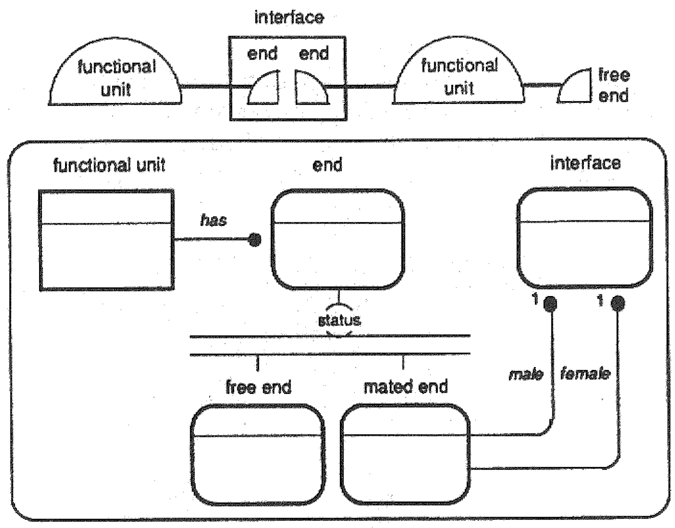
A Technical Solution may specify, in its turn, a number of (sub) Functional Units. This principle supports the decomposition/aggregation mechanism [3]. The resulting hierarchical structure constitutes a tree.



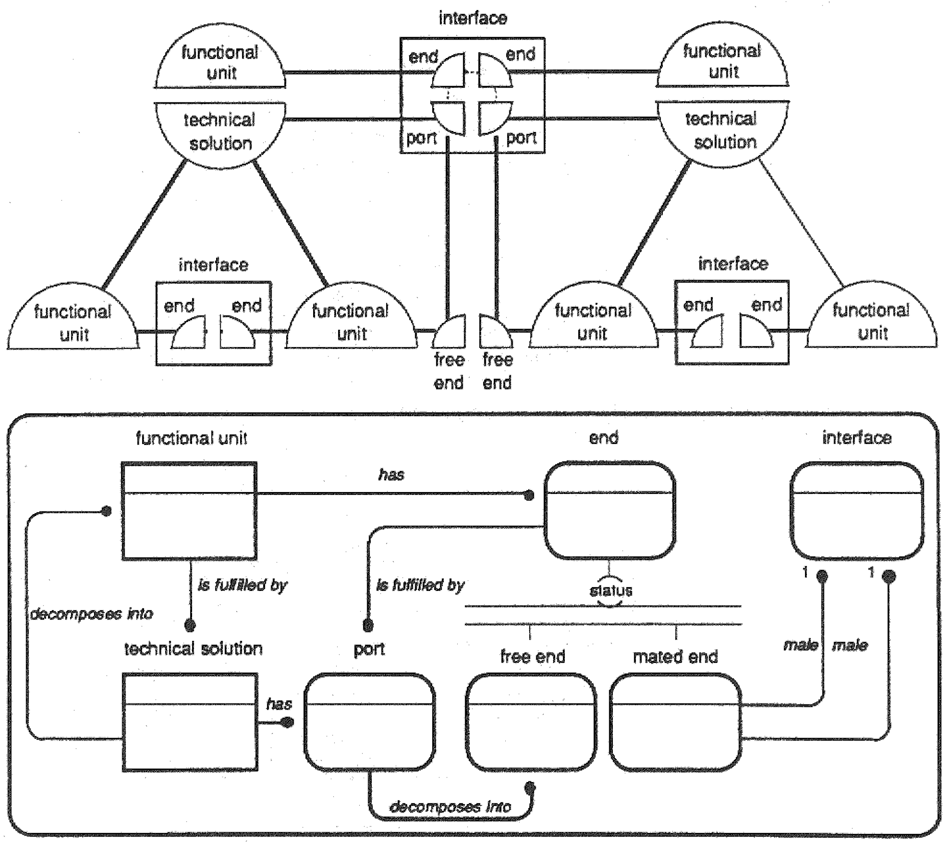
The Functional Units which constitute a decomposed Technical Solution are related to each other in a functional network. Each Functional Unit refers to a Node in the network. Each Node has zero, one or more Ends, and two Ends can be connected via an Interface. This allows us to define any kind of network relation between Functional Units.
Often the Functional Units and their interrelations will have geometrical aspects, which can be mapped on a geometrical/topological model. Such a topological model may also be interpreted as a (topological) network.
In this paper I will demonstrate how to combine the functional network and the topological network by an intermediate structure we have called Meta-Topology.

Functional Network

For historical reasons and because of the dominance of geometrical/topological relations in the modeling area which is covered by AEC products the functional network was based originally on a geometrical/topological structure.
A geometrical/topological structure acting as a functional network has several disadvantages. The most obvious disadvantage is that all relations have to be modeled according this point of view, even if a relation is not at all of a geometrical/topological nature. Another disadvantage is that the model could not support the use of different geometry/topology-descriptions for different applications at the same time. Gradually consensus arose that the kernel of the AEC Reference Model should stay entirely independent of any geometrical/topological specifications.
The result of this insight reflects in a pure type of functional network. According to this network a Functional Unit may specify a number of *Ends*. The End concept offers its Functional Unit a potential relationship with another Functional Unit (it may form an interface). The relation itself is established by an *Interface*, which connects two Ends of two different Functional Units on the same level in one branch of the hierarchical tree. Mark that an End may or may not be mated with another End by means of an Interface. Most parts and (sub)-assemblies will have free Ends.



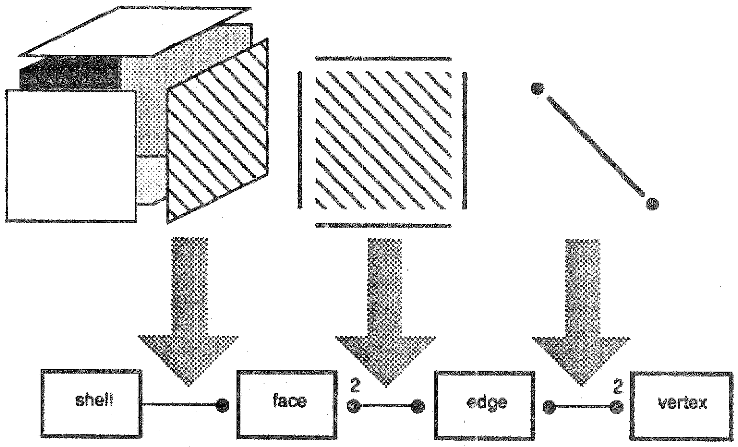
To solve the problem of the loose Ends, the dual concept Functional Unit versus Technical Solution is copied to the Ends: the functional part by the entity End, the technical part by the end of a Technical Solution called *Port*. After selecting a Technical Solution for a Functional Unit each End should be connected unambiguously with a Technical Solution Port in a one to one relation. If the Technical Solution decomposes into one or more sub Functional Units each Port should similarly decompose into one or more sub Ends. Those Ends must be in a free status.



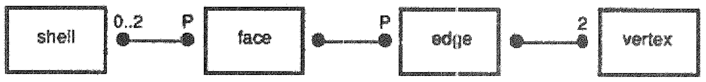
Generalized Topology

If we consider different topology frameworks for geometric modeling, a set of basic boundary elements is shared in almost every scheme. These basic boundary elements are vertex, edge, face and shell. Each of these boundary elements is attached to a geometry domain of a distinct dimensional order. A shell encloses a three-dimensional area. A face encloses a two-dimensional area. An edge for a one-dimensional area, and a vertex is attached to a singular

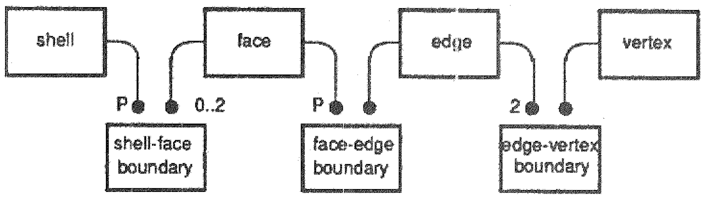
(nought-dimensional) point. Of course, the lower order dimensional areas reside in the same three-dimensional modeling space.
If we focus on these four basic boundary elements, we will notice that in most topology frameworks higher order boundary elements are defined, directly or indirectly, by lower order boundary elements. Neglecting all kinds of particularities in the different frameworks, we could generalize this principle in the following pure form:



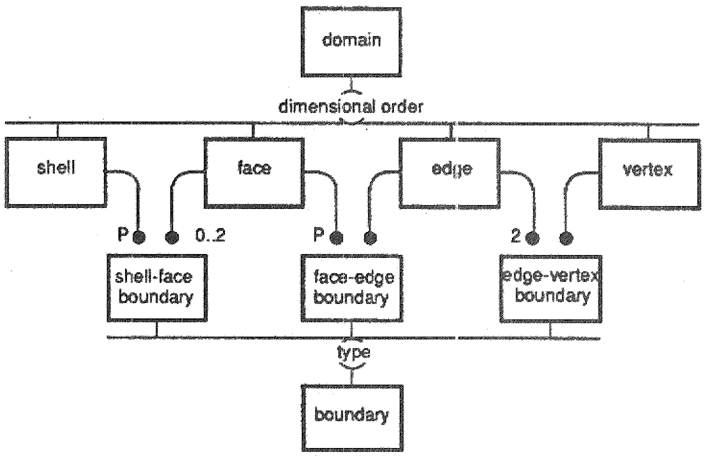
With this scheme one is able to represent solids without voids or holes and, in general, without depressions or protrusions. This 2-manifold scheme can easily be extended to a more general n-manifold scheme:



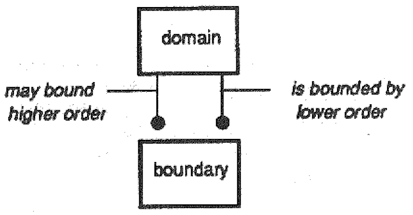
In the IDEF-1x terminology these relations are considered unspecific and need a further refinement. The usual approach is to create in-between entities:



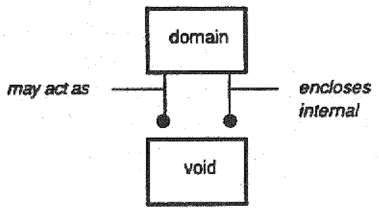
The basic entities and the in-between entities can be generalized using the category construction:



And by omitting the category entities:

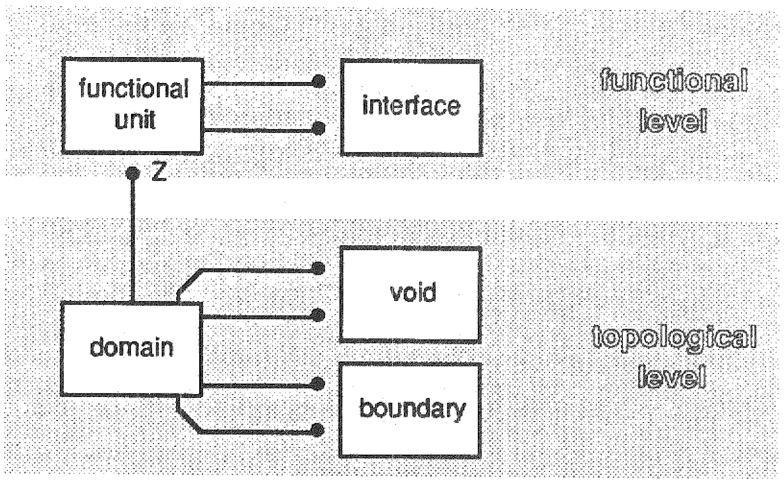


In this respect a Domain allocates an area in three-dimensional space of a certain dimensional order. It is therefore in itself not a boundary but it may act as (part of) a boundary of a higher order Domain. In reverse the boundaries of a Domain can be defined using lower order Domains. As noticed before, this schema is still too restricted to meet our modeling needs. What we lack is an additional structure to model holes in faces and inner spaces in solids. Besides we want to be able to define so called non-manifold relations: e.g. placing a vertex on an edge or a face, an option which is rather common in reference models and idealized models for certain analysis applications.
All these options can be offered when we generalize the Loop concept. In common topology the Loop is used to define faces. One loop is used to define the outer boundary while a number of successive loops define inner boundaries: holes. This principle can also be achieved by the relation a face-in-a-face. More general a Domain-in-a-Domain. The inner Domain may have the same or lower dimensional order and must reside completely within the enclosing Domain. As a result we are able to model a face-in-a-face, an edge-in-an-face, a vertex-in-a-face, etc. The diagram below shows the general idea.



Reference Topology

Why consider topology frameworks in this manner? The principle motive is the need for a reference topology. A reference topology is not in the first place meant to accurately describe the shape of an object, but to offer a geometrical/topological network to organize the information resources which constitute a product model. Therefore boundary elements, which represent the contours of a solid in a geometric modeling application, may become more significant by having also meaning in itself.
E.g. let's model, in an AEC application, a room in a building. For the topological specification a reference is made to a shell (third order Domain). At the same time a set of faces, edges and vertices is introduced, which are necessary to define the shell correctly. Information concerning this room can be stored by referring to this reference shell. If there is also information concerning a wall, which bounds this same room, it is obvious that we will use an existing face, which was already created to define the shell, to refer to. It is essential to realize that this face has a twofold function:
1 a topological function for partly defining a shell, and
2 a reference function for the object it stands for: in this case a wall.
In the same example information about the connection of two walls, e.g. a column, may refer to an edge, while information concerning the connection of a column and a beam can be attached to a vertex.
To avoid confusion because of the double semantics it is useful to split the data model into two levels: a topological level to describe a framework which has a pure topological nature, and a functional level for elements and relations which have also a functional significance. Contradictions should be avoided by recording each relation only once.



The IDEF-1x diagram above is meant to visualize the two levels of semantics. It is too simple and unrestrictive for practical purposes. However, the complete IDEF1x diagram is appended to this paper.
For reference topology the difference between material and non-material areas is not very relevant, at least on the topological level. To return to the room-in-a-building example: the reference shell for the room is, of course, void, or at most filled with air. Even a bounding face need not be materialized at all, for example if the room represents an open kitchen. In all cases these interpretations are decided at the functional level, not on the topological level.
Still the void concept is very valuable, also for reference topology. There is certainly a need to mark sub-domains, but the semantics should be more flexible than the choice between material or non-material! To have the the void concept available at the functional level is much more flexible and powerful for product modeling applications.
On the functional level two Functional Units are related by means of an Interface. Interfaces can now be distinguished into the Boundary Interface for a functional relationship of type Domain A bounds/is bounded by Domain B, and the Domain Interface to model a functional relationship of type Domain A encloses/is enclosed by Domain B.

Meta-topology

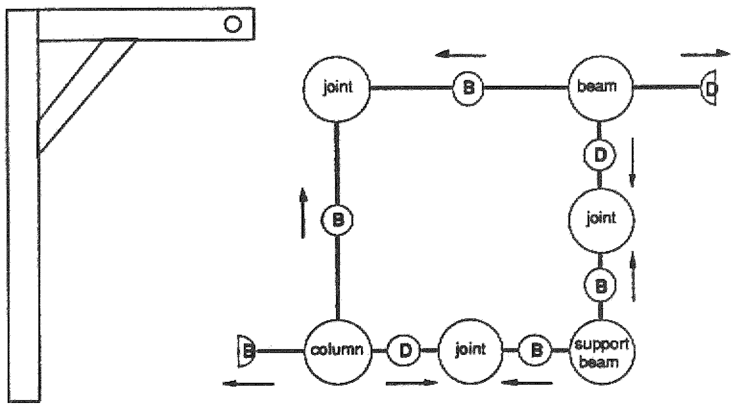
In fact we have created a three level structure now: on the top level Functional Units and Interfaces which form, if they are not further specified in a topological sense, a pure non-topological network. On the bottom level Domains, Voids and Boundaries, which form a pure topological framework. The in-between level links the functional network and the topological framework to each other: the meta-topology layer.
When we define an Interface to be the connection between two Functional Units this Interface need not be present all the time, e.g. the two separate parts of a door lock. Another possibility is one Functional Unit which interfaces with different partner Functional Units, e.g. a photo camera and its lenses. Yet another possibility is an Interface with a still unknown Functional Unit, e.g. a wall plug and the future use by an electrical apparatus.
In all those cases the two sides of the Interface have to be modeled separately. To be able to do this the *End* concept was introduced. Besides the reasons mentioned before, Ends are needed in a product model structure which is organized by a hierarchical tree, which is the result of decomposition or aggregation, depending the modeling approach: top-down or bottom-up. If such a tree structure is applied, it is inevitable that relations are cut, which exist between the branches of the tree, and have to be diverted along the branches to a common node [6].
It is obvious that the End concept on the functional level must have counterparts on the other two levels. This offers the additional advantage to restrict the modeling domain to possibly meaningful models.
On the meta-topological level Ends are categorized on their type (Boundary End or Domain End), on their direction (Boundary End: pointing to decreasing or increasing dimensional order, resp. Internal Boundary End and External Boundary End; Domain End: pointing to sub-domain or super-domain, resp. Internal Domain End and External Domain End), and status (free or mated by means of an Interface). Each combination has a restricted set of allowed relations.

On the topology level the *Side* and *Region* entities are introduced. Sides and Regions guarantee that relations are topological (Boundary or Void) or meta-topological (Boundary Interface or Domain Interface), but not both. They also fulfill the free End concept. The complete IDEF-1x model has been appended to this paper, while the table below shows the corresponding entities on the different levels. The entity names are slightly different to integrate this model with the General AEC Reference Model [1], especially one should read Functional Unit for Node.

network	meta-topology	generalised topology	topology			
node	node	domain	vertex			
			edge			
			face			
			shell			
end	boundary end	side	vertex side			
			edge side			
			face side			
			shell side			
	domain end	region	vertex region			
			edge region			
			face region			
			shell region			
interface	boundary interface	boundary	edge boundary			
			face boundary			
			shell boundary			
	domain interface	void				vt/vt
					ed/ed	ed/vt
				fc/fc	fc/ed	fc/vt
			sh/sh	sh/fc	sh/ed	sh/vt

Example

This example shows a simple frame structure which could be a Technical Solution for a gallows or a fixation for a sign-board. The Technical Solution can be decomposed into six (sub) Functional Units: *column, beam, support beam* and three *joints*. The functional network shows six Interfaces (four Boundary Interfaces and two Domain Interfaces) and two free Ends. The column has one free Boundary End to fix it to a foundation structure, while the beam uses a free Domain End to be able to connect the sign-board (or the criminal) to the frame. Both foundation and sign-board are Functional Units which are no members of this functional network. N.B. The arrows denote the direction of decreasing dimensional order.



Literature:

1. Gielingh, Wim, General Reference Model for AEC Product Definition Data, TNO-IBBC, August 1987, BI-87-87

2. Lee, Kunwoo and Gossard, David, A hierarchical data structure for representing assemblies, Computer Aided Design, Jan/Feb 1985

3. Tolman, Frits, c.s., A Modeling Space for STEP, TNO-IBBC, June 1988, PU-88.07-I

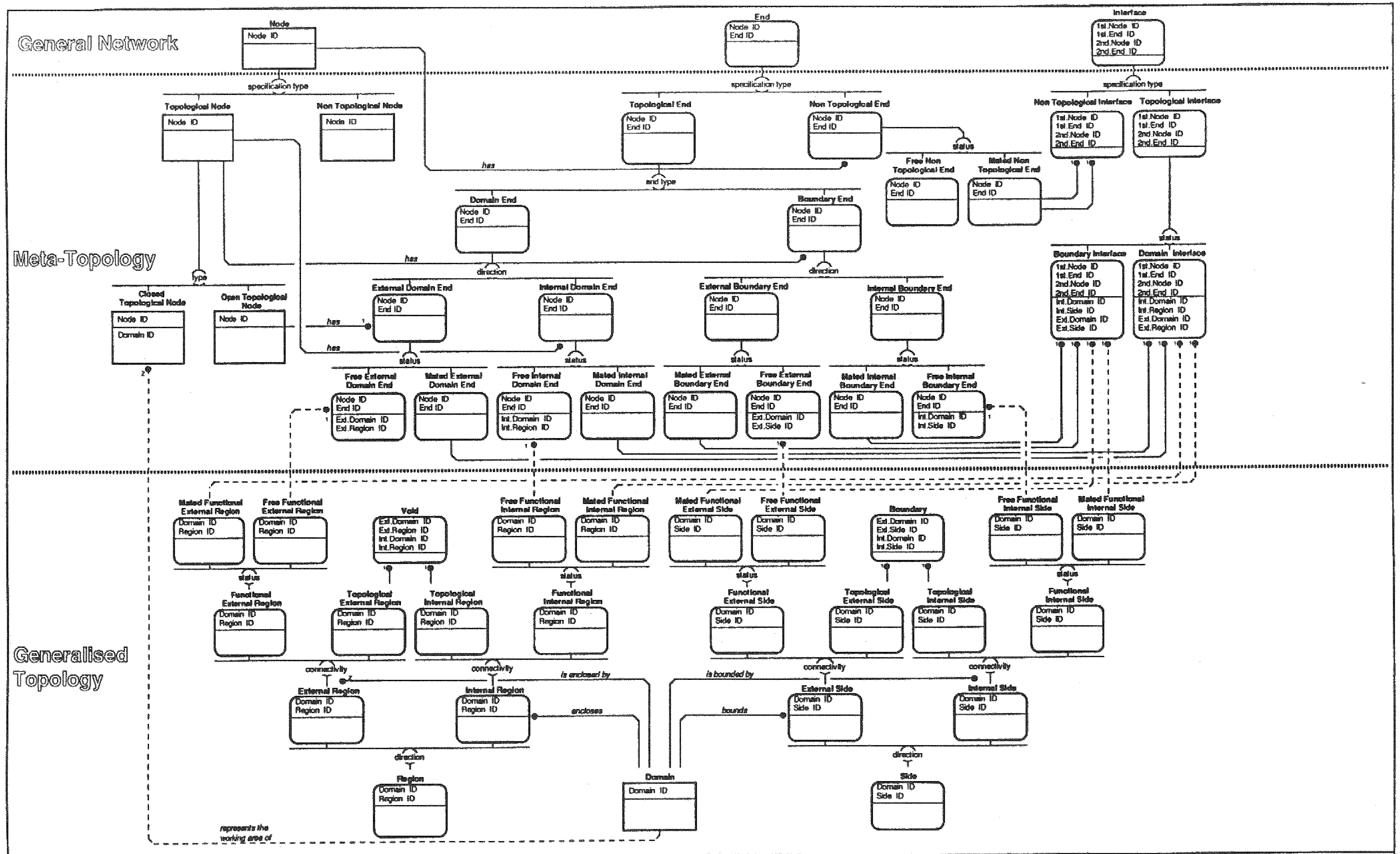
4. Weiler, Kevin, Two Taxonomies for Geometric Modeling Representations, 5/1/87, General Electric, Corporate Research and Development, Schenectady, NY 12301

5. Weiler, Kevin, Non-Manifold Geometric Boundary Modeling, Draft Prepared for SIGGRAPH '87 Advanced Solid Modeling Tutorial, 5/7/87, General Electric, Corporate Research and Development, Schenectady, NY 12301

6. Willems, Peter, A Functional Network for Product Modeling, TNO-IBBC, July 1988, PU-88-16

7. Willems, Peter, A Meta-Topology for Product Modeling, TNO-IBBC, July 1988, PU-88-10-II

8. Wilson, Peter, Euler Formulas and Geometric Modeling, Computer Graphics & Applications, August 1985



Author: Peter Willems
Date: October 3, 1988
Scope:
Status: Working

ISO TC184/SC4/WG1
General Reference Model for AEC Product Data

C.1. Meta-Topology

Reader:
Date: