# PDI TOOLS IN THE COMBINE PROJECT

Wim Plokker[1)] and Godfried Augenbroe[2)]

[1)]Researcher at TNO Building and Construction Research
Department of Indoor Environment, Building Physics and Systems
P.O. Box 29, 2600 AA DELFT, The Netherlands
[2)]Senior Researcher at TU Delft, Faculty of Civil Engineering
P.O. Box 5048, 2600 GA DELFT, The Netherlands

**Abstract**

*Results from the COMBINE project (Computer Models for the Building Industry in Europe) are presented. This EU-funded project, now in it's second phase, deals with the development of integrated building design systems.*
*COMBINE has concentrated on providing efficient data exchange between a number of "design nodes" in a network, each node being identified as an actor at a workstation in a LAN connected design team. The paper deals with the way data exchange facilities are delivered to the project in the form of a customizable interface kit.*

## 1 INTRODUCTION

The initial phase of the EC project COMBINE (1990-1992), was a first step in the development of an Integrated Building Design System (IBDS) through which energy services, functional and other performance characteristics of a planned building can be analyzed (Augenbroe, 1993). COMBINE1, with the co-operation of 14 European groups, successfully produced an Integrated Data Model (IDM) which was capable of interfacing with six design tool prototypes. The realised data exchange facilities rely on present product data technology and are compliant with the STEP standard (ISO-STEP, 1992; 1993).
With the groundwork successfully completed, the follow-up COMBINE2 ('93-'95) project seeks to combine the tools developed into operational prototypes, according to functional specifications resulting from building projects in practice. The existing range of interacting design tools is being extended to cover building regulations, costing, product catalogues and architectural and HVAC CAD tools.
The resulting system will be primarily configured to support multi-actor data exchange in specific architectural and engineering design office settings.
An extended shape description is provided in the IDM in order to support operations on the shape of the design artifact and exchange spatial information about it with other actors.
We will discuss the progress of the tasks in the present COMBINE phase.
In particular, we will deal with the way that the actual data exchange is accomplished by employing state of the art PDI (Product Data Interchange)



technology. The resulting COMBINE PDI tools enable two-way exchange between each design tool function at a design node in the system and a central node which acts as the central Data Exchange System (DES). The DES core contains an OODB implementation of the conceptual integrated building mode (IDM). The actual exchange of data is accomplished either through STEP files (off-line design tools) or through on-line data exchange.
Figure 1 shows the implementation architecture of the targeted IBDS prototypes.

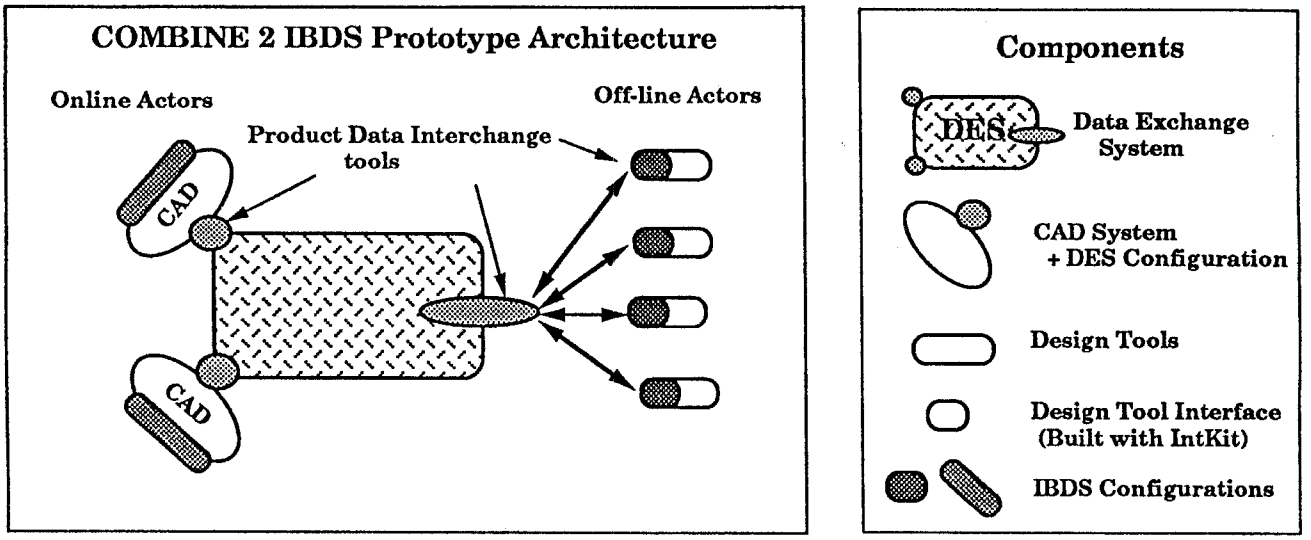


Figure 1 Implementation Architecture of COMBINE IBDS

The following sections will deal with the PDI tools that enable the off-line STEP file exchange on the design application side.
The focus will be on the development of the new COMBINE Interface Kit that enables rapid development of STEP-interfaces by design application builders.
After the Interface Kit has been tested and validated by the application interface developers in the COMBINE project it is expected to become a key deliverable to be used by external application builders and other R&D projects.

We will mainly report on the development of a STEP file processor as part of the node interfaces consisting of a parser and a mapping module.
The new version of the interface kit which is being developed in the present second COMBINE phase has a number of extended features, compared to the first release of the interface kit in COMBINE1.
The major aim is to enable rapid development of design tool interfaces.
It will be shown how design tool developers can use the new interface kit as a state of the art PDI tool to build and customize interfaces.

## 2 COMBINE INTERFACE KIT

PDI (Product Data Interchange) facilities within the COMBINE project are accomplished by enabling two way data exchange between each node and an OODB implementation of the conceptual building model (IDM). The actual exchange is accomplished either through STEP files or through on-line data exchange (ISO STEP Part 21, 1992). The interface kit, discussed here, deals with the off-line communication accomplished through the exchange of STEP files. Besides the basic functionality of reading and writing STEP files the interface kit also deals with viewing and selection capabilities embedded in a GUI. Users will thus be able to view the complete model and make part selections for handling by the design tool, e.g. a particular storey or a building zone. The data exchange kit is based upon the EXPRESS modelling language as meta description of the data and STEP neutral file format for the actual file exchange (ISO STEP Part 11,1993). The new version of the interface kit, being developed in the present second COMBINE phase, has a number of extended features, compared to the interface kit which was a key deliverable of the first COMBINE phase (Plokker, 1993). These extensions concern three major areas. The core of the interface kit is an SDAI C++ binding implementation for general handling of the exchanged information (ISO STEP Part 22,1993). Generic mechanisms for communication with the user of a design tool are built on top of this core. They are supported by view and selection modules, in which representation knowledge of IDM entities is incorporated, thus enabling graphically supported selections. Another part focuses more on support of the interface development process and provides a library for geometrical calculations and topological search operations on the IDM to facilitate mapping between instances of the IDM and design tool Aspect Models. The major aim is to enable rapid development of design tool interfaces.

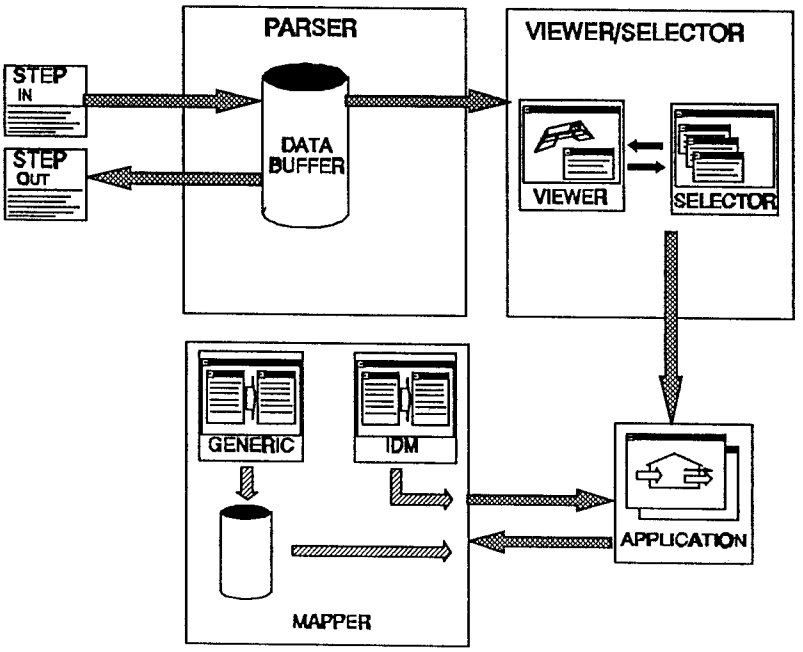


Figure 2 The three basic components of the interface kit.

The three major functions depicted in figure 2 are discussed in detail in the next paragraphs.

## 2.1 THE SUBSCHEMA EXCHANGE MECHANISM

In COMBINE a two way data exchange takes place between the "design nodes" and an OODB implementation of the IDM in a "central node". Each application only needs a restricted set of both entity definitions and entity occurrences. For instance a U-value calculation does not need information about the HVAC systems. Moreover, applications do in general not deal with all instances in the building model at once. For instance an overheating risk assessment application would normally only operate on a limited set of rooms.
The case of a restricted set of entity definitions is handled through a subschema mechanism, which is discussed more in detail in this paragraph. The case of a restricted set of entity occurrences is solved by equipping the interface with selection capacities, which will be discussed in section 3.

To build an interface between an application and the DES, an application developer will start by making a data model for this application. In COMBINE, this model is meant to express the external data requirements of the design tool; it is called the design tool's Aspect Model. By comparing an aspect model with the more generic IDM, a minimal subset of IDM entities and relationships can be determined which is needed to derive the aspect model. In this fashion, IDM-subschemas for the input and output data are derived from the complete building model, by ruling out all the entity definitions and attributes which are not needed by the application. Additional rules and constraints may be added to a subschema, e.g. specifying constraints on instances. This is to be compared to traditional WHERE-clauses in database queries. In this way it is possible to limit the set of exchanged instances to those that actually make sense to a particular design tool. The subschemas are fixed by means of EXPRESS descriptions. For communication between an application and the cental data model four EXPRESS schemas are needed :

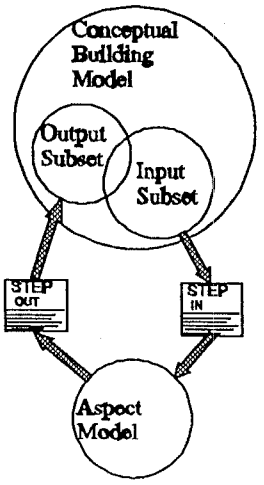


Figure 3 The four models needed for interfacing.

- The conceptual building model (the generic IDM)
- The Aspect Model (application-specific)
- An input subschema (application-specific IDM subschema)
- An output subschema (application-specific IDM subschema)

The two latter schemas form the bases of communication between an application and the IDM. The subschema mechanism was chosen for both a technical and a conceptual reason. Most design applications considered in COMBINE run on a PC-platform and are not capable of managing large complete IDM instances for complex buildings. The conceptual reason is that working with subschemas forces an application developer to explicitly state the data requirements for his application. The completeness of the IDM can thus be checked. It also opens the way to a much more intelligent building design system where information exchange is monitored by a supervisory module which can fire actions on the basis of the information contents of the exchange files. The COMBINE Exchange Executive (ExEx) which is part of the DES is devised to perform these type of control functions over the data exchange events.

## 2.2 SDAI PARSER

An interface, built with the parser kit, consists of three basic parts. The first part is a plain parser which is capable of reading STEP Physical File Format. This part reads in a STEP file and puts it into a data repository. Data can be retrieved from this repository via access functions. The structure of the repository and application specific access functions are generated from an EXPRESS file containing the schemas for the exchanged information.

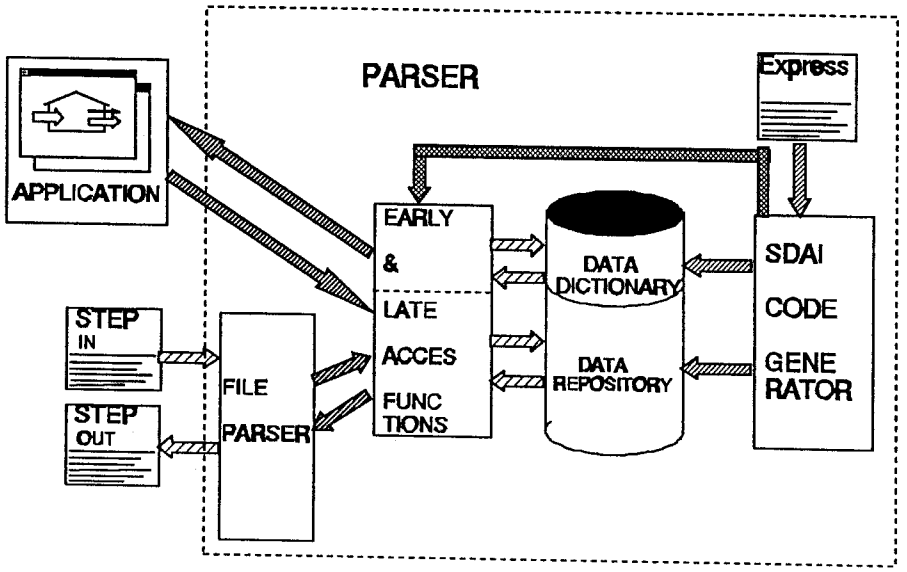


Figure 4 Basic architecture of the SDAI compliant parser.

The parser kit itself consists of two basic parts. A software library providing general functions to read and write a STEP-file, to access the data repository and a program generating C++ source code from EXPRESS files. The generated source code are the elements with which an application designer can then proceed to build a specific interface.
The access functions and the generated code are compliant with the SDAI (Standard Data Access Interface) C++ binding specifications which are described in part 22 of ISO 10303.

SDAI offers the following functionality :

- Access to and manipulation of data entities which are described using the EXPRESS data specification language.
- Access to the EXPRESS definitions of all data elements that can be manipulated by an application process.
- Validation of the constraints defined in EXPRESS at the discretion of the application process.

The main reason to use SDAI within the COMBINE project is the use of the data dictionary. An overview of the dictionary schema of SDAI is given in the figure below.
The data dictionary stores information about the schemata used by SDAI. Because these schemata are defined in EXPRESS, the structure of this dictionary model reflects the structure of EXPRESS itself.

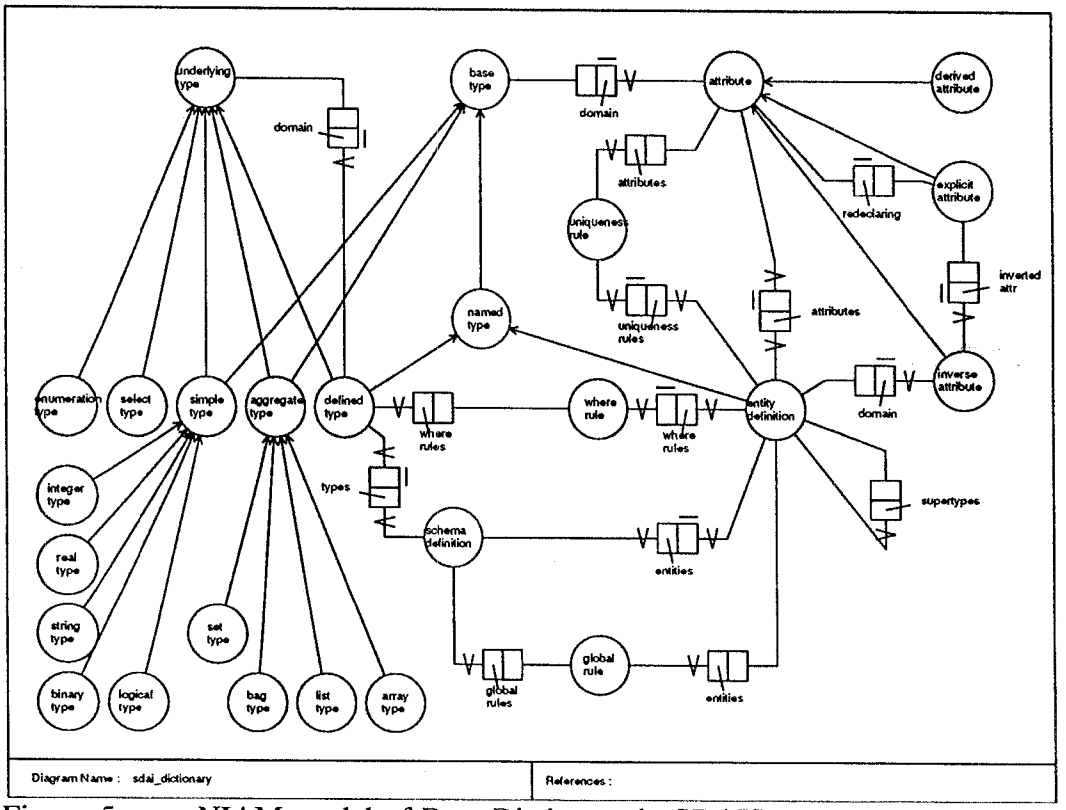

Figure 5 NIAM model of Data Dictionary in SDAIS.

The use of SDAI offers several advantages.

- An application is isolated from the data, this allows considerable additional flexibility when enhancing both application and database features.
- An SDAI implementation will let an application programma access and manipulate data of which the structure is described in EXPRESS. The programmer does not need to know how the data is stored (in a file, working form or in a database).
- If an application interface is build using SDAI, an easy switch between several SDAI implementations from different vendors can be made.

### 2.2.1 Early and Late Bindings

Two types of programming language bindings are specified: those independent of the EXPRESS schema being implemented and those dependent on the EXPRESS schema being implemented. These types are referred to as "late" and "early" bindings, respectively, of the functional specification to a particular language. Both bindings are supported within COMBINE.

Early bindings are specific in nature and are applicable only to applications based on a specific EXPRESS schema. References to particular EXPRESS constructs are made via embedding the name of the construct directly into function or subroutine names. In this case, the binding does not specify the complete set of functions available to the application programmer, but does supply guidelines how these functions should look like. The binding specifies any functions not dependent on the EXPRESS schema being implemented in the standard and a mapping algorithm for describing the functions that are dependent on the schema being implemented. For example, an operation to set the value of attribute a1 of entity b1 is described as SetValueb1a1("value").

Late bindings are generic in nature and one binding is suitable for application development regardless of the context of the application or the EXPRESS schema being implemented. All references to particular EXPRESS constructs are made via parameters passed as input via language interface. Using this type of binding, it is possible to write EXPRESS driven applications that provide generic functions for any schema. In this case, the binding specifies the complete set of subroutines or functions available to the application programmer. For example, an operation to set the value of attribute a1 of entity b1 is described as SetValue(a1,b1,"value").

The advantage of early binding over late binding is that type checks are done at compile time, where for late binding checks have to be done at runtime.
Late bindings defined within SDAIS concern :

- IsKindOf functions
- Testing of attributes
- All dictionary functions

The new interface kit supports both early and late bindings.

# 3 INTERFACE KIT EXTENSIONS: GRAPHICAL BROWSER AND SELECTOR

As pointed out in the paragraph concerning subschemas, most design tools do not operate on the whole building at once, but only on a part. A good example of such a design tool is an overheating risk assessment method, where calculations are only performed for the most critical locations. How to determine the most critical locations, is the skill of the user of such a design tool. The decision which rooms to choose for an assessment is made by the local user at the DT node and not by the central data exchange system (DES). The DES will send all relevant entity occurrences that are in the subschema of the application. A selection of which rooms have to be calculated must be made on the side of the design tool. To support this process of selecting specific entities to operate upon, the interface kit is extended with graphical viewer and selector modules.

The graphical selector kit is aimed at both the DT-developer and DT-user as end users. The developer uses the functionality offered in the graphical selection toolkit to build a communication dialogue with the user of a design application.
A set of generic classes will support the building of the graphical user interface(GUI) whereas the behaviour of the selectors can be configured by the DT designer. When a need for a selection on a specific object is identified, the developer is able to implement a dialogue in which the DT user is only able to select an object of the specified type within a specified range.
The viewing and selecting modes are thus set up in a predefined way through which the DT user is supported. He is able to have a graphical view of the data he is acting on, while adding or changing data in the SDAI repository.
The functionality of the graphical selection tool is such that it is able to generate a representation of any "physical object" in an IDM-instance. In order to be able to do this the viewing and selection modules must have knowledge about the geometry and topology model of the IDM. The knowledge inside the viewing model can however be kept to a minimum if the building model is well structured.
Of particular relevance is the physical-object entity in the IDM. It is used to describe physical "things" that are or can be produced by a natural process or can be manufactured (van Nederveen, 1993). Examples are: a building, a door, a wall, any part of a building, any relevant object in the environment of the building. In COMBINE, spaces are considered as physical objects. Every physical object has three relationships. The first describes its decomposition, the second its connections to other physical objects, the third is a reference to representation items.

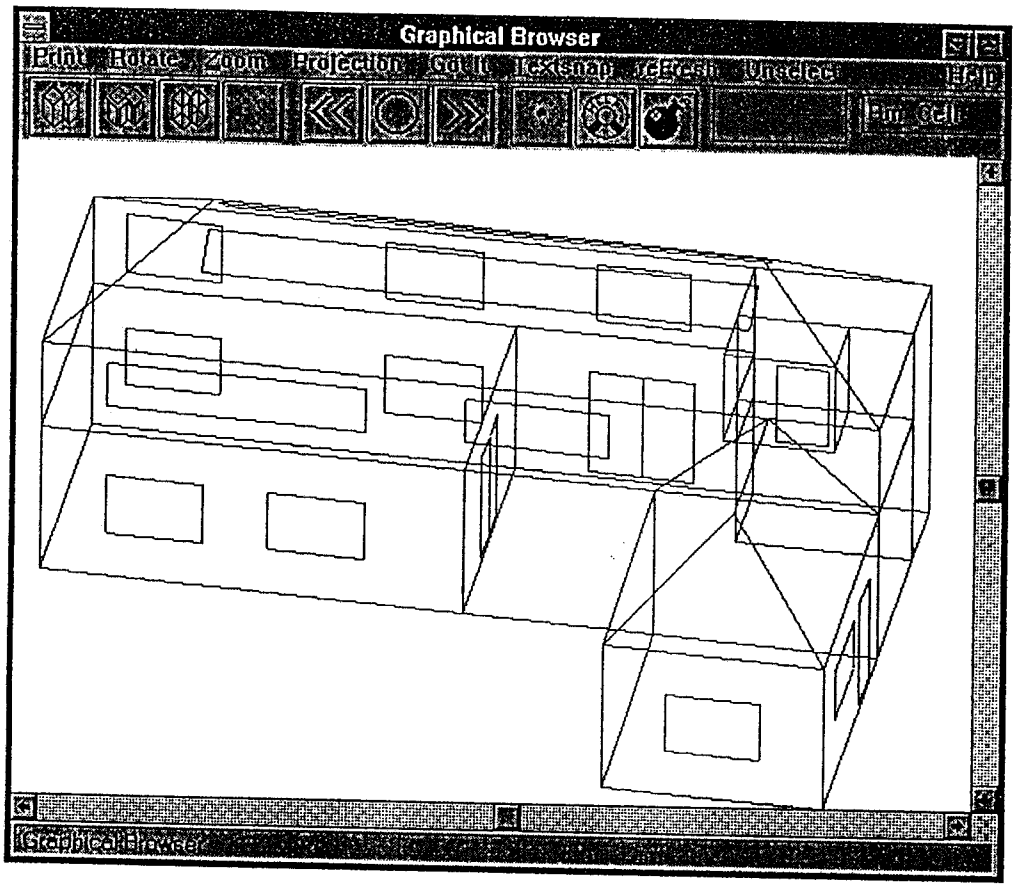


Figure 6 Example of a selection window

A representation item is an element of product data that either gives a representation of a physical object using a single representation technique, or helps to define another representation item. This construct is based on Part 43 of ISO 10303.

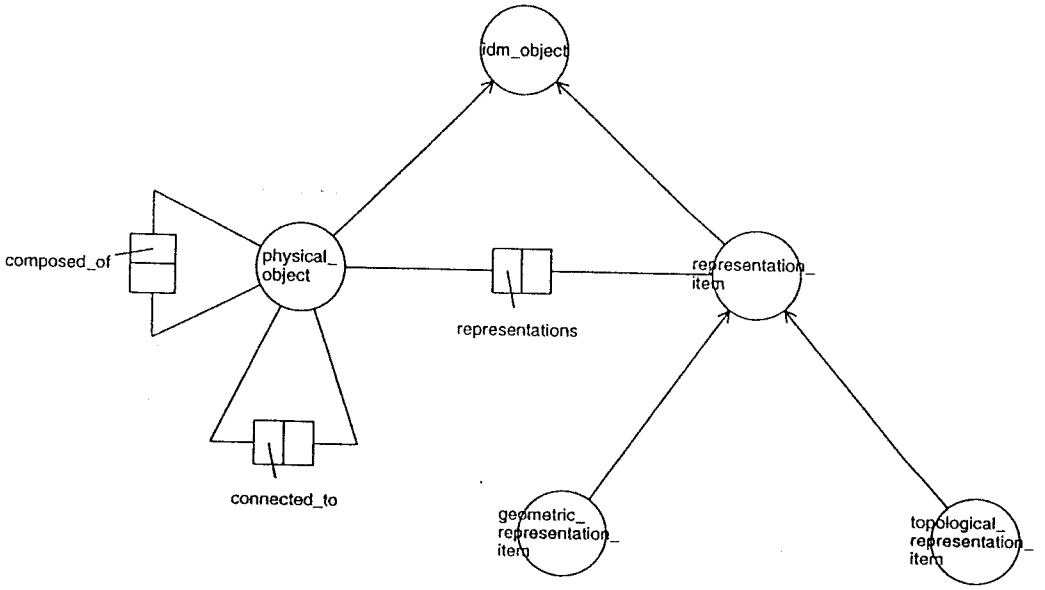


Figure 7 NIAM schema of the top level structure of the IDM.

The representation technique used in COMBINE is RM-rep (Relation Model representation), which is well suited for shape descriptions of building objects. RM-rep is more or less based on B-rep, and was developed to better fulfil the needs of the building industry (Augenbroe,1994).
The viewer and selector modules have knowledge about the decomposition and representation structure of the IDM. This knowledge is based on classification of relations, the knowledge about RM-rep is hard coded in the modules. The modules know how a cell is constructed from faces and how faces are constructed from edges, etc.

## 4 MAPPING FACILITIES

In COMBINE the IDM development is driven by the design tool aspect models and communication needs, whilst trying to keep the IDM as generic as possible in order to guarantee completeness (within a well defined scope) and easy expendability (beyond the present scope).
Design tool aspect models will thus in general not be subsets of the Integrated Data Model. It is all but trivial to map IDM entities to aspect model entities and vice versa. While the physical exchange of the information is fully automated by use of present STEP technology, the mapping between different models is still a hard job. An example of a mapping needed by many applications in COMBINE is the mapping between the RM-Rep entities in the IDM and the area of a wall. Although this sounds easy, it is a rather complex mapping. The mapping starts with the entity wall which can be decomposed in segments. Each segment has a *rm_face* as representation_item. Through *rm_loop, rm_edge_side, rm_edge, rm_edge_end* and *rm_vertex* the actual coordinates of the point which make up the face can be reached.
There are some languages which are capable of describing mappings between two EXPRESS schema's. But they are not capable of modelling the above complex mapping. Even if there existed such mapping languages, their use in combination with the SDAI repositories would require the same effort as building a programming language compiler. For this reasons we follow a practical approach within COMBINE. Those mappings common to different design tools are identified. These mappings will be implemented as a class library, which acts on the SDAI repository. The use of a data dictionary as offered in SDAI facilitates this job to a large extent.

## 5 FUTURE DEVELOPMENTS / CONCLUSIONS

The different part of the interface kit are still under development at this stage of the project. A first release of the SDAI core is tested by the design tool developers at this moment. Constraint checking, cardinality rules and other validity rules are not fully implemented yet, but both early and late binding access to the data repository are fully supported.
The development of the viewing and selection modules is nearing completion. While testing the SDAI core of the interface kit the design tool developers will define their mappings with respect to the aspect models. After an analyses of most wanted mappings a library of common mapping routines will be built.

In COMBINE phase 1 the use of an interface kit proved to be essential for the implementation and actual use data exchange based on conceptual models. In COMBINE phase 2 the use of SDAI as the basis of a revised interface kit facilitates the interfacing between an application and the integrated data model even more. Especially the data dictionary can be of great use for implementing a general mapping mechanism. The use of subschemas of the IDM facilitates the implementation of interfaces and keeps the information flows manageable. The extension of the interface kit with viewing and selection modules helps the interface builder to develop and customize interfaces with a rich functionality in an integrated engineering environment

## 6 REFERENCES


Augenbroe, Godfried (1993). COMBINE Final report. CEC-DGXII Report.

Augenbroe, Godfried (1994).Extended Topology in Building Design Systems. First Congress on Computing in Civil Engineering (Vol. 2), ASCE, Washington, Khalil Khozeimek (ed.), pp 1196-1203

ISO-STEP Part 11 (1993), ISO-TC184/SC4 Industrial Automation Systems - Product data representation and Exchange - The EXPRESS Language Reference Manual. International Standard

ISO-STEP Part 21 (1992), ISO-TC184/SC4 Industrial Automation Systems - Product Data Representation and Exchange - The Step File Structure. International Standard

ISO-STEP Part 22 (1993), ISO-TC184/SC4 Industrial Automation Systems - Product data representation and Exchange - Standard Data Access Interface Specification. Draft International Standard Nederveen, G.A. van (1993), IDM+ Subset. COMBINE work Report, TNO-Bouw, Delft

Plokker, Wim (1993), EXPRESS/STEP interface kit for COMBINE. COMBINE work Report, TNO-Bouw, Delft