GENERALISED MODEL SUBSET DEFINITION SCHEMA

Matthias Weise, Peter Katranuschkov and Raimar J. Scherer
Dipl.-Ing, Dr., Prof. Dr., Institute of Applied Computer Science in Civil Engineering, University of Technology Dresden, Germany
{Matthias.Weise, Peter.Katranuschkov, Scherer}@cib.bau.tu-dresden.de

SUMMARY

The idea of a shared product data repository that can be efficiently used in distributed concurrent engineering environments takes practical shape with the ripening of the IFC platform. However, structured methods for defining and deriving different partial model views that can easily be used by domain-specific applications are not yet available.
This paper presents a generalised model subset definition schema (GMSD) which tackles many issues related to partial model retrieval and the use of domain-specific views. The schema, defined in EXPRESS for consistence with the IFC project model, contains two subparts: (1) constructs supporting the dynamic selection of object instances in model server queries, and (2) constructs enabling the development of view definitions that can be used as permanent part of a model server environment, to support the automated derivation of discipline-specific model subsets for architectural or structural design, facilities management etc. Discussed are the requirements that have governed the development of GMSD, its basic concepts, a prototype server/client implementation and first test-bed results. The latter indicate the great potential of the suggested structured approach for the definition of appropriate model subsets to improve the amount and quality of the exchanged data between actors and applications.

INTRODUCTION

The successful application of concurrent engineering and project collaboration methods are among the most important factors for competitive advantage in the construction industry today. It is widely accepted that these new methods of working can be best supported using product data technology as basis. The necessary framework for that purpose, i.e. a well-defined set of standardised inter-related data models, is provided by the current IFC platform [Wix & Liebich 2001].
The vision of IFC is to define a standard in which all disciplines involved in the life cycle of a construction project can add their specific data to a common shared model and then use that data in the context of their specific requirements and tasks. This approach has many advantages. However, it also requires continuous maintenance of the model data consistency. Given the complexity of the IFC project model this proves to be a difficult problem.
For example, a direct exchange between users and applications cannot fulfil the IFC vision as it practically enforces sequential data processing with only poor, mainly manual consistency checking. Therefore model servers that can maintain and manage a shared product data repository are required. However, as indicated by many researchers, in this case too a number of problems have to be dealt with, such as long transactions, heterogeneous software applications, domain views and mappings, access rights, versioning etc.
In a narrower scope, questions related to the data access to the shared repository include:
– the amount of necessary and sufficient data,
– the exchange format (typically SPF [ISO 10303-21 1994] but increasingly replaced by XML or by direct RPC methods in advanced applications), and
– the communication paradigm used.
Whilst the second and the third are more 'technical' issues, the first is a conceptual problem.
In principle, the amount of needed data can range from individual attributes to full models but for practical purposes most important is the support for *partial, domain-specific model views*.
The advantages of using partial models are manifold. They can provide support to domain applications that are not capable to 'understand' the full global model, but they can also dramatically reduce the quantity of the exchanged data and hence the network traffic, provide well-defined or even standardised reliable domain views and simplify the data consistency and coordination problems. Thus, the adequate processing of partial models is a major prerequisite for effective product data management.

This paper presents a generalised model subset definition schema (GMSD) which is dedicated to the tackling of the data access aspects outlined above. GMSD enables the realisation of client/server or file based transactions in a structured manner, at different levels of granularity, and for different data exchange formats.

RELATED WORK

With the progress of the IFCs, the idea of a common shared product model begins to take practical shape in the AEC landscape. However, there has not been much work yet to address the use of the IFC platform as a true enabling data repository for distributed project environments. Development of IFC model servers is up to now mostly limited to the access of individual objects and/or full model export, eventually subject to a more or less fixed set of pre-selection conditions. A simple and easy to use structured method to define and derive different partial model views is not yet available.

Indeed, SDAI [ISO 10303-22 1998] provides a well-defined API to access EXPRESS-based model data, but it is not fully suitable for advanced client/server infrastructures. SDAI operations are on a very low level of granularity. Partial model retrieval enabling off-line work within long transactions is not supported.

In the area of database development useful hints provide the object-oriented query languages OQL [Cattel et al. 2000] and SQL-3 [Melton 2002]. However, they are not directly applicable for the management of product data models since (1) some specific requirements are not sufficiently covered, and (2) they both rely heavily on the underlying database models (ODMG and the relational model respectively), instead of using a true product model language. They also do not address the specification of views in the line of the layered approach of the IFC platform [Wix & Liebich 2001].

Early solutions dealing with partial models within the AEC domain have been proposed in the EU project COMBINE 2 [Augenbroe 1995, Lockely & Augenbroe 2000]. In COMBINE an integrated data model (IDM) is used as a repository from which the needed subset of information is derived. A sub-schema approach is suggested using the same modelling language like the IDM for defining a model subset. Additional constraints are used to further minimize requested data on instance level. This approach is quite powerful but it does not serve well dynamic (ad hoc) view specification and usage. It is also not so well suited to support IFC applications which directly operate on the IFC model schema.

Most closely related to the work described in this paper is the *Partial Model Query Language* (PMQL) developed by Yoshinobu Adachi [Adachi 2002]. PMQL borrows many concepts from SQL. It is capable of using SQL statements and is in fact strongly optimised for implementations with underlying relational databases. The language is specifically designed to support client/server interactions, providing straight-forward language bindings for SOAP and XML. However, PMQL normally requires many request-response cycles to retrieve the information for a specific view. It does not contain dedicated constructs for explicit view specification and has some problems with path expressions, nested queries and inheritance hierarchies.

REQUIREMENTS TO A FORMAL MODEL SUBSET DEFINITION

In the basic concurrent engineering process, with the typical steps of long database transactions i.e. (1) check out, (2) local data manipulation, and (3) check in followed by data merging, a *generalised model subset definition* can be efficiently used for step one and, additionally, to support step three, so that a highly generic, self-controlled solution with as few as possible user interactions can be achieved. It must be able to provide the necessary specifications to retrieve the requested product data, as well as appropriate additional constructs that can be used to reintegrate already processed (and modified) data.

Within step one it must *define the data structures* needed to specify a partial model request. Specifically, as the IFC model is defined in EXPRESS, its realisation should be built also upon the EXPRESS specification [ISO 10303-11 1994] and the underlying object-oriented paradigm. Additionally, it must have the capacity to be *externalisable* into different syntactic forms such as XML, XML schema, SPF, or other serialisation forms for object-oriented structures, and it must be *embeddable* into various high-level communication protocols, as part of standard methods like 'get', 'put', 'search', 'inspect' etc. Therefore it should *not* be a data manipulation language but should rather focus on the specification of model subsets using a neutral representation format.

Accordingly, the requirements to the model subset definition schema (GMSD) are set up as follows:

– support requests for *arbitrary data sets* of EXPRESS-based object-oriented data structures using predefined and/or ad hoc selection mechanisms,
– support requests on *different levels of granularity* allowing to sort out any irrelevant information for the specific view of interest,
– *minimize request-response cycles* by providing concepts for recursive and nested queries as well as one-step tackling of aggregation and inheritance hierarchies,
– enable the realisation of a *simple declarative language* using only a small set of basic view definition concepts, i.e. focus only on what is to be done and not on how it is to be implemented,
– support the use of *predefined views* such as the Coordination View provided for IFC2x (see http://www.bauwesen.fh-muenchen.de/iai/ iai_isg/doc/IAI-ISG-88.htm), and finally,
– be *widely applicable* with different data exchange (XML, SPF, …) and communication techniques (HTTP, SOAP, RMI-IIOP, CORBA, …) to enable application-independent API implementations.

Out of scope are considered questions related to any specific communication paradigms, the reintegration of partial models into a common global repository, the explicit specification of methods and operations, and more comprehensive transaction management including login, session support etc. Thus, the generalised model subset definition schema can be seen as an add-on to basic technologies such as SQL and SDAI that does not contradict to but can substantially improve standard data exchange techniques.

In the following sections, the conceptual design of GMSD, its basic structure, a first prototype implementation and early results of its usage are presented and discussed.

THE GMSD SCHEMA

GMSD is designed in the spirit of known extended object-oriented languages for database environments, such as *ObjectStore*, *GemStone* and others. However, it is specifically oriented to the support of EXPRESS-based models, with special attention to IFC. Also, unlike any of the aforementioned data manipulation languages, GMSD is *not* a language but a *schema* which allows a neutral definition format with possible mappings for various practical data exchange and server/client realisations. It is comprised of two subparts which are almost independent of each other with regard to data but are, at the same time, strongly inter-related within a coherent overall process. These two parts are: (1) *object selection*, and (2) *view definition*.

As the name suggests, the first part is purely focused on the selection of object instances using set theory as baseline. The second part is intended for post-processing of the selected data in accordance with a specific partial model view. Such views can be completely predefined (or instantiated) to support standard domain/discipline perspectives, correlated with the respective IFC extension models.

Figure 1 below presents the top level entities of the GMSD schema and illustrates the envisaged method of its use in run-time model server environments. Note, however, that even though one of its major goals is the definition of partial model views, GMSD itself is highly generic. It offers a minimal set of constructs that are largely independent of the specific data model used and it does not provide any statement about the semantically meaningful data from engineering point of view.

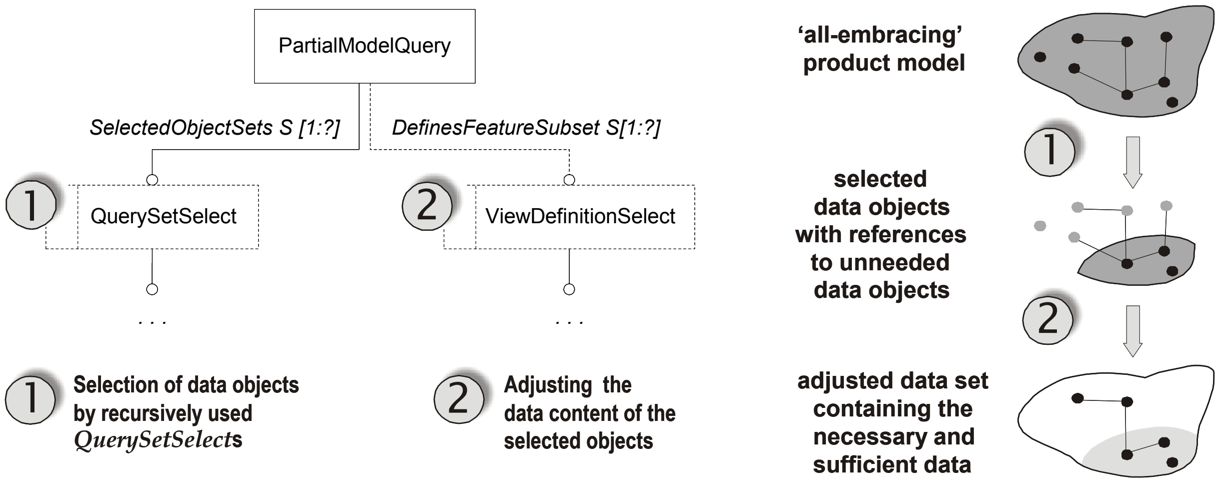

PartialModelQuery
SelectedObjectSets S [1:?]
DefinesFeatureSubset S[1:?]
1
QuerySetSelect
2
ViewDefinitionSelect
. . .
. . .
1 Selection of data objects by recursively used QuerySetSelects
2 Adjusting the data content of the selected objects
'all-embracing' product model
1
selected data objects with references to unneeded data objects
2
adjusted data set containing the necessary and sufficient data

Figure 1 Top level structure of the GMSD schema and the related instance level operations

Dynamic object selection

Object selection is introduced by the top level construct *QuerySetSelect* (see figure 1). Its components provide basic set theory operators, path expressions and nesting facilities to support the retrieval of instance model subsets *dynamically*, as part of model server queries. This serves two objectives:

– enable object filtering in accordance with different criteria, and
– reduce the set of actually relevant data for a specific view *before* applying any view-dependent operations on class level, as defined within the second part of the schema.

The latter is similar to the typical projection/selection of database tables performed before using a set operation such as *join* to provide a specific view on the data.
Figure 2 below shows the main entities defined in the object selection part of the GMSD schema.

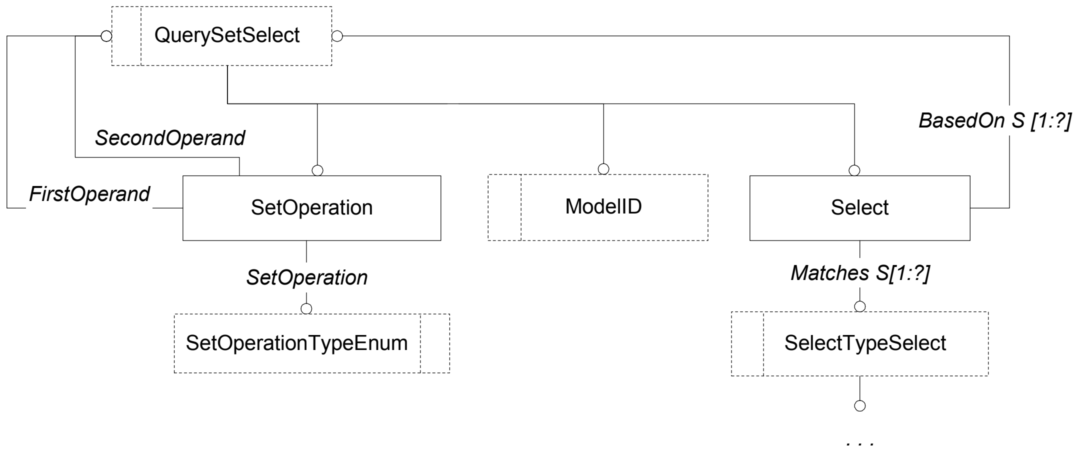


Figure 2 Top level entities for dynamic object selection in the GMSD schema

The entity *QuerySetSelect* offers three possibilities to select objects:
(1) by the *ModelID*, defining all objects belonging to a specific model version,
(2) by the result of various *selection type constructs,* such as object instances filtered by the value ranges of one or more attributes, all instances of a specified type eventually including traversal of its subtypes, referenced entities, presence/absence of certain references to other objects in the selection set – typically resource objects such as *IfcMaterial* –, and so on (not shown on figure 2 for brevity), and
(3) via the *set operations* union, difference and intersection, applied to the result from (1), (2), or another set operation.

A *PartialModelQuery* instance defines via its attribute *SelectedObjectSets* all selected objects which can be identified by the *ModelID*, by instances of the entity *Select* or by *SetOperation*.
ModelID is a string which serves for initial selection of all instances of the model with that *id*. Therefore at least the *id* of the model must be known to start a request sequence. The managing of model *id*s is not in the scope of GMSD but it is a natural requirement that should normally be fulfilled by any model server. For example, in IFC the unique entity *IfcProject* can be used for that purpose.
The identification of the model of interest is the basis for all further selection conditions specified by the attribute *BasedOn* of one or more instances of *Select*. If a condition provided by the attribute *Matches* is satisfied by an object referenced via *BasedOn*, then it belongs to the result set of the *Select* statement. As mentioned above, by reference to *SelectTypeSelect* several selection constructs are available. The structure of the GMSD schema allows to extend these easily by more sophisticated constructs, or to adapt the schema more closely to the needs of IFC.

View definition

The second part of the GMSD schema is dedicated to its capability to provide view definitions. Unlike object selection, view definitions are intended to be developed *in advance*, e.g. as part of a particular model server environment, to support various discipline-specific applications (such as structural analysis and design, building services design, facilities management, cost estimation etc).
A view definition is specified on *class level*. Typically, it would be applied after object selection. It can be used to:

– sort out object instances by their type or reduce their data content on attribute level,
– sever references to unneeded objects to create a consistent data set that is semantically more meaningful for the specific requesting application,
– generalise objects to supertypes, to lower the complexity of the resultant model subset,
– extend or reduce object selections by including/excluding referenced objects in accordance with some specified criteria.

Not supported are mapping operations that may be needed in cases when the target schema is structurally different. For that purpose, mapping languages such as EXPRESS-X [Denno 1999] or CSML [Katranuschkov 2001] may be used as appropriate.

Figure 3 below shows the elements of the GMSD schema supporting view definitions.

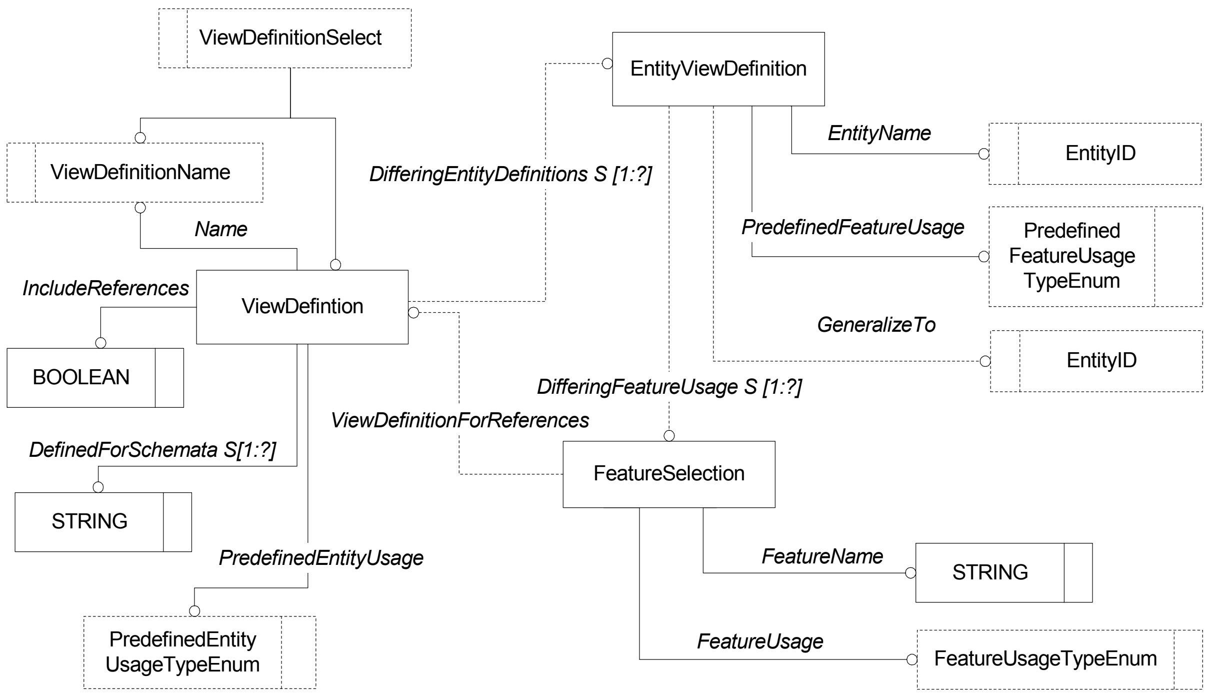


Figure 3 Entities for the specification of partial model views in the GMSD schema

The top level class *ViewDefinition* provides a unique name to identify predefined views, a Boolean flag indicating if referenced entities should be included by reference or by value, the names of the schemata for which the view definition is valid, a predefined entity usage and its overriding extensions. With the *PredefinedEntityUsageTypeEnum* a default handling of object types can be set to (1) use all object types with all their features, (2) use all object types with only those features that are needed to keep data integrity, or (3) omit all object types and rely only on the extension specifications. Detailed definitions can then be provided by a set of *EntityViewDefinition*s enabling the handling of each object type on feature (attribute) level. Furthermore, *FeatureSelection*s can be specified to 'fine-tune' the usage of each feature, whereby the available feature usage types (use, conditional use, conditional delete) can be applied multiple times. Such alternative definitions can be assigned to referenced objects to allow for different interpretations, depending on the actual reference paths. This can be useful in cases where the reference path is important to process the actual object instances properly. For example, in the IFC model the processing of the object geometry provided via *IfcRepresentation* can be given in dependence of the element type (*IfcBeam*, *IfcWall*) using the particular referenced geometry objects.
Additionally, through the attribute *GeneralizeTo*, an object type can be typecast to one of its supertypes, to enable a more general representation that may be required by the requesting application.

PROTOTYPE IMPLEMENTATION

GMSD has been developed in conjunction with the client/server environment of the iCSS project, supporting the management of EXPRESS-based data models [iCSS 2002]. Additionally, a dedicated client fully based on GMSD has been implemented for testing purposes and as a pre-processor for applications in the scope of iCSS and other projects [Stickl 2002].

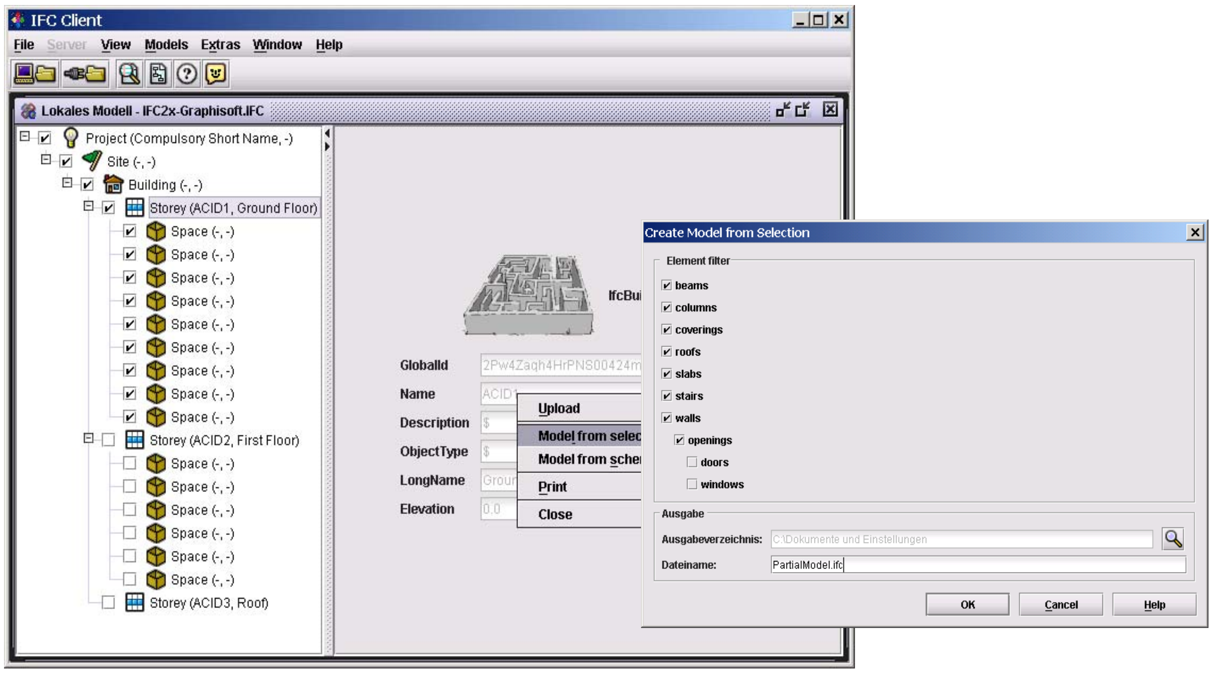
On the server side all incoming requests are generically handled, regardless of the particular model schema. On the client side all model requests are defined via GMSD, using the Information Container API suggested in [Katranuschkov 2001] on top of Java RMI. The client implementation is specifically adapted for IFC2x to enable efficient retrieval and visualisation of different views of the whole model, selected buildings, or individual storeys. The results of a query are written in SPF format, in conformance with the functionality supported by most legacy applications.

Within this prototype implementation three main steps can be typically identified for each model query. Assuming that a structural view is requested for a particular building storey, e.g. for slab dimensioning, these three steps are as follows:

(1) Select the model version of interest using a server-specific function – normally, this will be the newest available version.

(2) Issue a request for all high level instances of the selected model version using GMSD. In the discussed example this will be the respective *IfcBuildingStorey* container. This is needed to provide an initial selection set that will then be used to restrict all further requests to instances of the selected storey.

(3) Issue a GMSD-based request for objects belonging to the selected container (i.e. the *IfcBuildingStorey* instance) using a specific structural view definition to extract the objects needed for the respective structural engineering task.



GlobalId: 2Pw4Zaqh4HrPNS00424m
Name: ACID1
Description: $
ObjectType: $
LongName: Grour
Elevation: 0.0

Element filter: ✓ beams, ✓ columns, ✓ coverings, ✓ roofs, ✓ slabs, ✓ stairs, ✓ walls, ✓ openings, ☐ doors, ☐ windows

Ausgabeverzeichnis: C:\Dokumente und Einstellungen
Dateiname: PartialModel.ifc

Figure 4 Screenshot of the GMSD-based product model client

FIRST RESULTS

At this stage, GMSD has been tested on several examples from IAI/IFC using relatively simple view definitions. All tests showed satisfactory results with respect to performance and the resultant model sets. However, as they involved relatively small models, more practical case studies are yet needed to estimate correctly performance aspects, especially in view of some anticipated 'expensive' operations.

A representative example based on an IFC file exported from a sample ArchiCAD drawing is discussed in table 1 below. The geometry of the used model – a small 3-storey buildings –, is provided ön figure 5.

Test 1 in table 1 presents the result for the original source model, checked in at the iCSS model server, then directly retrieved by the GMSD client without any filtering conditions, and imported back into ArchiCAD to visually check correctness (round-trip test). The other tests use this base model applying different GMSD selection and filtering options.

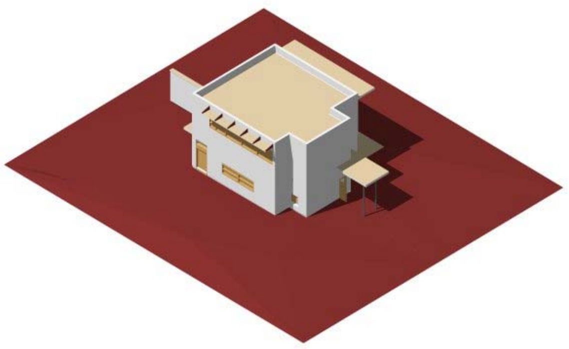
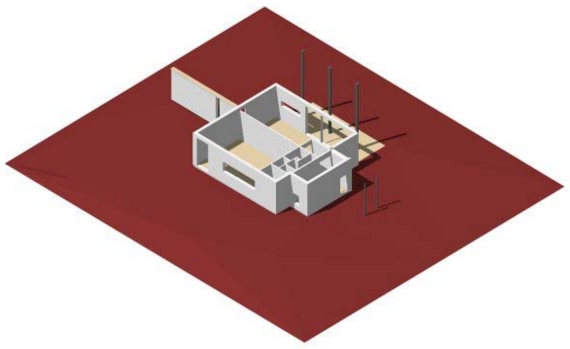



Figure 5 3-D view of the presented test example (*left*: full model, *right*: partial model of ground floor)

Test	GMSD specification	Result set	In % of full model	Comments
1	Selection of all objects, no filter	10013 instances (-456)	96%	Some instances disappeared, probably due to 'unreachable' objects (dead links). Thus, even here GMSD may help to obtain more consistent and eventually reduced model subsets.
2	Selection of first floor, no filter	9344 instances	79%	The partial model included much more instances than originally expected (approx. 1/3 of all tangible data + control objects). The reason for that was found to be in the large number of doors and windows 'carrying' a huge amount of references to resource objects, especially for geometry definition.
3	Selection of ground floor, no filter	6375 instances	61%	Clearly smaller model than in test 2 due to the smaller number of doors and windows.
4	Selection of ground floor without *IfcSpace* objects, no filter	6375 instances	61%	No difference to test 3 → space elements are automatically included by reference to warrant consistency.
5	Selection of ground floor, no doors and windows	3861 instances	37%	Great difference to test 4 (+ 24%) by sorting out all doors and windows.
6	Selection of first floor, no doors and windows	5095 instances	49%	Filtering out all doors and windows has even greater impact than for the ground floor (+40%).

Table 1 Selected test results from the use of GMSD in the implemented client/server environment

From the tests performed so far the following observations can be drawn:

- By smaller models the reduction of the resultant data set ranges from 10 to 60%, greatly depending on the number of building elements with complex geometry.
- The size of domain view models will depend in a similar way on the amount of geometry data used. Thus, whilst with building services or facilities management only slight data reduction may be expected, in models for cost estimation the data may be pruned to as much as 80-90%. In particular, for the structural analysis model developed in the IAI ST-4 project with participation of the authors [IAI ST-4 2002], a more detailed examination showed possible reductions of 40 to 50%.
- In future practical models including all discipline domains, the amount of exchanged data with discipline-specific applications may drop up to 80-90%, however with some probable performance overhead. Having in mind that a full model file can easily be 1+ GB, this is a clear practical benefit.

CONCLUSION

In the preceding sections a new structured approach for the tackling of partial models and domain-specific views was outlined. This approach has the following advantages:

– it enables the definition of views in conceptually clear manner, suitable for standardisation,
– it is not coupled with a specific communication paradigm or data exchange format which provides for easier application integration and implementation of pre-processors and plug-ins,
– it enables domain applications to concentrate on their own business logic by shielding them from unnecessary model details,
– last but not least, it can substantially improve data exchange and sharing by the provided facilities bringing about strong reduction in the amount of exchanged data by guaranteed consistency of the resultant model subsets.

Some drawbacks of the approach are seen in certain performance concerns and in the small dependence on the correct handling of model ids by the used model server. These should be considered in future schema versions. Further efforts are also needed to extend selection / view definition operations by more sophisticated and more powerful methods.

ACKNOWLEDGEMENTS

The authors would like to thank the German Research Foundation (DFG) and the German Ministry of Education and Research (BMBF) for funding the projects “A Cooperation Model for the Control of Diverging Design Data” (DFG Sche 223/27-2) and “iCSS – Integrated Client-Server System for a Virtual Enterprise in the Building Industry” (BMBF 19W9949C) which have greatly facilitated this work.

REFERENCES

Adachi, Y. (2002) *Overview of Partial Model Query Language,* VTT Building and Transport / SECOM Co. Ltd., Intelligent Systems Lab., VTT Report VTT-TEC-ADA-12,
available from: http://cic.vtt.fi/projects/ifcsvr/tec/VTT-TEC-ADA-12.pdf

Augenbroe, G. /ed/ (1995) *Combine 2, Final Report* CEC-JOULE program, Brussels.

Cattel, R. G. G., Douglas, K. B. et al. (2000) *The Object Data Standard ODMG 3.0,* ODMG Publ.

Denno, P. /ed/ (1999) *EXPRESS-X Language Reference Manual,* ISO/TC184/SC4/WG11/N088, available from: http://www.steptools.com/library/express-x/n088.pdf

IAI ST-4 (2002) *IAI ST-4 Project "Structural Analysis Model and Steel Constructions"*, Volumes 0, I, II, III, available from: http://cib.bau.tu-dresden.de/icss/structural_papers.

iCSS (2002) *Integrated Client-Server System for a Virtual Enterprise in the Building Industry (iCSS) – Project Description*, available from: http://cib.bau.tu-dresden.de/icss/factsheet-en.html

ISO 10303-11 IS (1994) /Cor.1:1999/ *Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 11: Description Methods: The EXPRESS Language Reference Manual*, International Organisation for Standardisation, ISO TC 184/SC4, Geneva.

ISO 10303-21 IS (1994) /Cor.1:1996/ *Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure*, International Organisation for Standardisation, ISO TC 184/SC4, Geneva.

ISO 10303-22 IS (1998) *Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 22: Implementation Methods: Standard Data Access Interface*, International Organisation for Standardisation, ISO TC 184/SC4, Geneva.

Katranuschkov, P. (2001) *A Mapping Language for Concurrent Engineering Processes*, Diss. Report, Schriftenreihe des Lehrstuhls für Computeranwendung im Bauwesen, Heft 1 (Hrsg. R. J. Scherer), TU Dresden, Germany.

Lockley, S. and Augenbroe G. (2000) *Data Integration with Partial Exchange*, Proc. of International Conference on Construction Information Technology, INCITE 2000, Hong Kong, pp 277-291.

Melton, J. /ed/ (2002) *Information Technology Database Language SQL – Part 10: Object Language Bindings (for SQL:200n)*, ISO/IEC FCD 9075-10 / 32N0747.

Stickl, R. (2002) *Unterstützung der vernetz kooperativen Projektdatenverwaltung – Einbindung unabhängiger Ingenieuranwendungen in integrierte Lösungen auf der Basis des IFC-Projektmodells,* Diploma thesis (in German), TU Dresden, Germany.

Wix, J. and Liebich, T. (2001) *Industry Foundation Classes IFC 2x*, © International Alliance for Interoperability, available from: http://www.iai-ev.de/spezifikation/IFC2x/index.htm.