SISIP - A Systems Integration Platform based on Distributed Persistent Objects

Arne-Jørgen Berre, Frode Høgberg, Magnus Rygh,
David Skogan, Jan Øyvind Aagedal
Systems Integration Technology
Department for Informatics
SINTEF
P.O.Box 124, Blindern
0314 OSLO, NORWAY
E.mail: Arne-Jorgen.Berre@si.sintef.no
Phone: + 47 22 06 73 00
Fax: + 47 22 06 73 50

Abstract

Integrated environments in general may support the four integration areas: data-integration, control-integration, presentation-integration and process-integration. The SISIP architecture takes a unified object-oriented approach to these four integration areas. The underlying infrastructure is based on distributed persistent objects, realized by a combination of object-oriented database technology and distributed object technology. The SISIP object model, SIOM, provides a unified object model for all four integration dimensions. The object model combines a structural object model based on EXPRESS and the ODMG object model, with a behavioral object model based on the OMG Interface Definition Language. Modeling for SISIP-based systems is supported through the object-oriented SIMOD methodology, which is based on the OOram role modeling methodology. The SIDE development environment is itself SISIP compliant. The SISIP approach is being specialized for various domains, in particular GEOSIP for Geoscience applications, GISSIP for Geographical Information Systems, CIMSIP for Computer Integrated Manufacturing and BISSIP for Business Information Systems.

1. Introduction

The goal of SISIP Integration framework is to facilitate integration and interoperability among heterogeneous systems. SISIP extends from our earlier work on the COOP integration framework [Berr92, Berr93] by providing conformity and extensions to current technologies OMG/CORBA, ODMG and EXPRESS. This paper presents an overview of the different parts of SISIP. First, an introduction is given to the area of systems integration and interoperability based on our Double Toaster reference model. SISIP takes an object-oriented approach to the four integration areas shown through the reference model. The SISIP object model, SIOM, is introduced next, with its three languages SIODL Object Definition Language, SIOML Object Manipulation Language with mappings to C++ and Smalltalk, and SIOQL Object Query Language. The SIMOD modeling methodology, based on OOram, is used for the development of SISIP compliant systems. This is supported through the SIDE Development Environment, which itself also is SISIP compliant. SIMOD and SIDE is presented before a presentation of some areas where SISIP is applied. Finally there is a conclusion and description of future work.

2. Systems Integration and Interoperability

2.1 Goals of integration

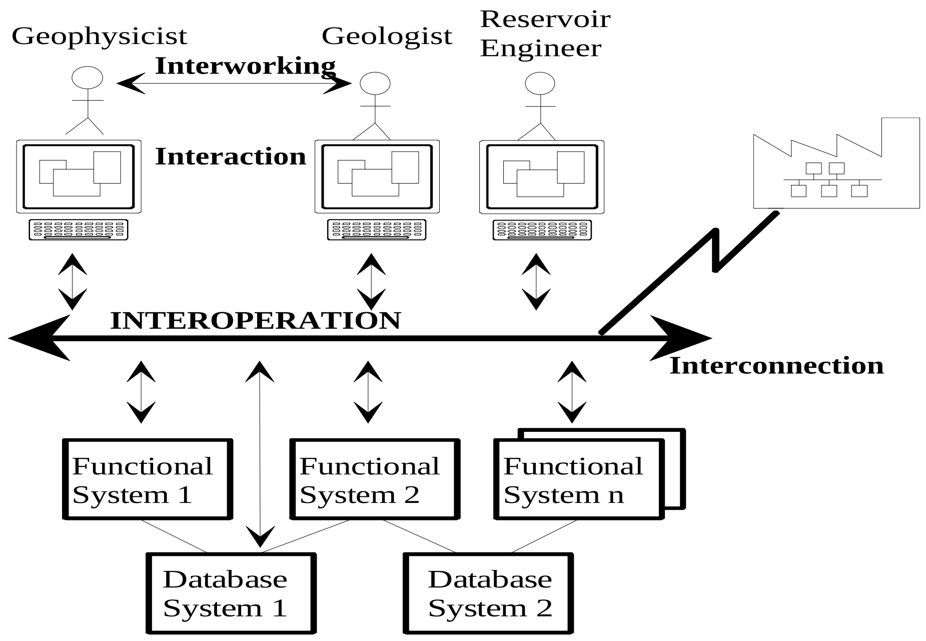


Figure 1 Integrated Environment for Geoscience applications

Many application domains, such as CAD/CAM, CIME, Geoscience (Oil&Gas Exploration & Production), Geographical Information systems and others, share the common goal of having integrated work environments. The requirements from geologists, geophysicists and reservoir engineers are to have an integrated environment that supports seamless integration between different tools. Figure 1 shows the need for supporting cooperation between users, supporting each user with tools for her tasks and for the overall working process, through a set of tools that can access enterprise-wide databases and system-services.

The environment should provide different tools for the different roles, to provide the functionality needed for the tasks of that particular role. It should provide support for cooperative work on the production of the field-evaluation-report, and it should guide the workprocess so that it follows the procedures and guidelines for a field-evaluation project. The vendors of applications in this area would like to have an environment where it is easy to utilize and reuse available functionality, so that the vendor can concentrate on their particular interest areas. It should be an environment supporting effective reuse, and easy use of services for data management, inter-application communication and user-interface construction.

The dictionary-definition of integration from [Webster] is:
Integration *Complete (imperfect thing) by addition of parts; combine (parts) into a whole. (lat. Integrare: make whole) The focus is on making the whole by combining the parts.*

The environment builder challenge is to facilitate the environment user view of an integrated environment by seamlessly "gluing" different underlying system parts together. "Gluing" implies a need for interconnectivity between the systems. Two or more systems are interconnected if they can exchange messages. This, however, only guarantees communication. In order to achieve integration it is also necessary with interoperability between the systems. Interoperability means mutual accessibility and usability of information representation, information equipment, information systems and users.

A definition of interoperability is given in [Brod92]
Interoperability: *Two components X and Y can interoperate (are interoperable) if X can send requests Ri for services to Y, based on a mutual understanding of Ri by X and Y, and if Y can similarly return responses Si to X.*

This means that two systems can interact jointly to execute tasks. What is required is a mechanism that facilitates interoperability between system components, by providing support for high level interaction among components. An object-interaction mechanism such as that provided by OMG/CORBA or COM/OLE is a suitable approach for this.

2.2 Double Toaster Reference Architecture

Systems integration can be viewed both from an environment user perspective and from an environment builder perspective. The environment user is concerned with the perceived integration at the environment's interface, while the environment builder is concerned with the feasibility and effort needed to achieve this perceived integration.

A taxonomy for problems and solutions for systems integration and interoperability has been developed at SINTEF, based on a refinement of the ECMA/NIST Toaster model with concepts from the European Eureka Software Factory architectural model.

Figure 2 shows a logical reference-model for systems integration, identifying 8 areas for integration. This reference-model refines the ECMA/NIST reference model for Software Engineering Environments [ECMA90,Thomas92] which describes services in 4 integration-areas: data-integration, control-integration, presentation-integration and process-integration. The ECMA/NIST model is refined by a distinction between multi-model and single-model data-integration, and a distinction between request-oriented and notification-oriented control-integration, a separation of the user-interface part and the functional parts of tools in presentation-integration, and a separation of user-oriented and group-oriented work-process integration.

Request-oriented control-integration: is the extent to which tools are able to interact directly with each other, by requesting and providing functional services. *Notification-oriented control-integration*: is the extent to which tools are able to interact by sending out notification about certain events. Other tools might register interest for being notified about these events. *Single-model data-integration:* is the extent to which tools are able to share common data and information that are stored and manipulated through one single data model and a corresponding storage service. *Multi-model data-integration:* is the extent to which tools are able to share common data and information that are stored and manipulated through multiple data models with corresponding multiple storage services. *Display-oriented presentation-integration:* is the degree to which a common look-and-feel is provided by the tools which are used. *Model-oriented presentation-integration:* is the degree to which the functionality presented through the display is accessed and combined from one or more underlying functional models. *Interaction-oriented process-integration*: is the extent to which the user's working-process and use of tools can be guided by a model of the work process and the methodology to be followed. *Interworking-oriented process-integration* is the degree to which group-work and interworking between different people is supported.

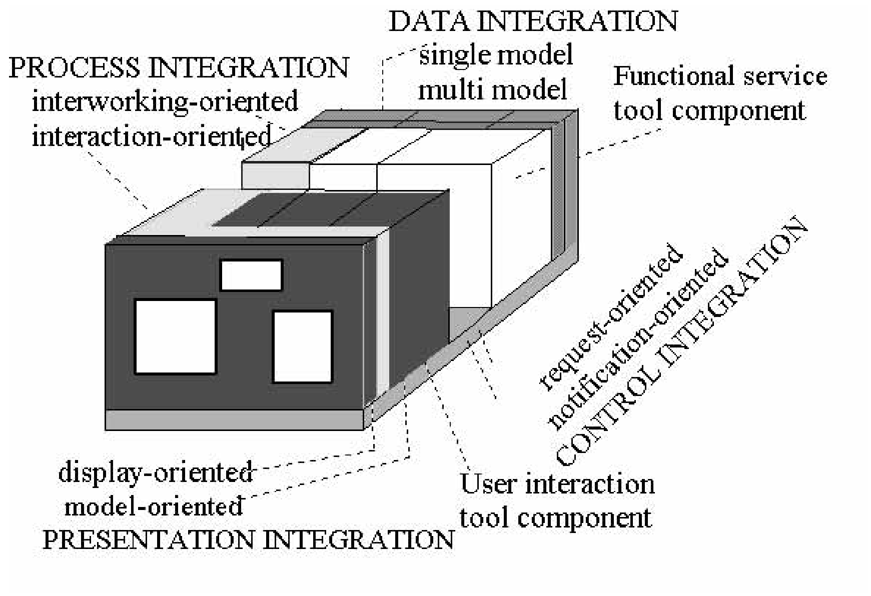


Figure 2 The Double Toaster - Refined ECMA/NIST Toaster Model

This integration taxonomy is an extension from earlier work presented in [Nils90] and is described in more detail in [Berr92, Berr93] . Our earlier work on Object-Oriented Technology, Object-Oriented Databases and Distributed Systems [Ber86, Ber88, ABM+90, Ber92] has lead to the belief that the object-oriented approach is feasible in all four integration areas.

3. SISIP Architecture

SISIP is an integration architecture based on an object-oriented extension the Eureka Software Factory project Software Bus architecture, that takes an object-oriented approach to the integration areas described in the Double Toaster reference model. The goal of SISIP is to address the various integration domains described in the Double Toaster reference model in a uniform way. An object-oriented architectural approach is taken to achieve this.

SISIP can be viewed as a Distributed Heterogeneous Object Management System with support for heterogeneous implementations for objects, and an object-model, SIOM, which unifies concepts from Distributed Systems, Database Systems and Object-oriented Systems.

The conceptual model for the application-programmer is to have a world of distributed persistent objects. It should be transparent that objects might represent information and functionality in underlying connected systems. Figure 3 shows the pool of distributed persistent objects which represents the totality of functionality and data available as objects. This gives a uniform support for both large-grained and fine-grained objects.
The figure illustrates the view that both components and items within the components might be viewed as objects.

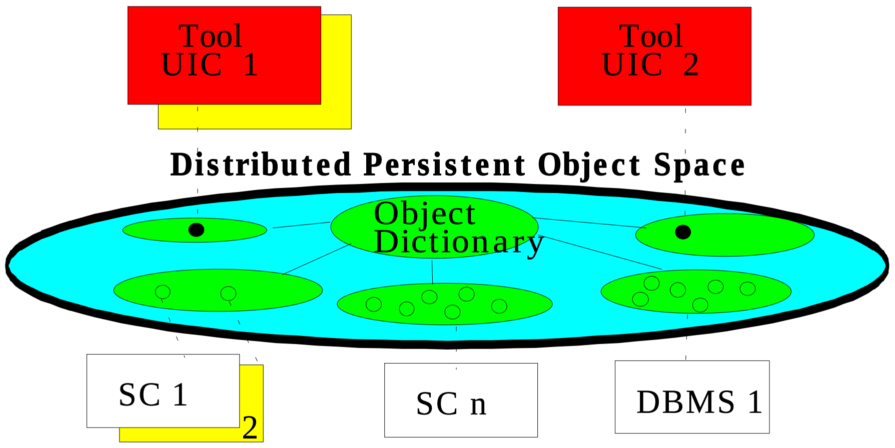


Figure 3 Integration through a Distributed Persistent Object Space

The notion of distributed persistent objects represents an integration of concepts from three different technology-areas: Distributed Systems, Database Systems and Object-oriented Systems. The integration of these three technologies is done through SIOM, SISIP Object-Oriented Model, in a way that give support to all the different integration-areas:

Request-oriented control-integration: through viewing existing systems as objects with a certain interface. *Notification-oriented control-integration:* through a service for event-registration and event-notification for objects. *Single-model data-integration:* through viewing existing data-items within components and database-systems as objects. *Multi-model data-integration:* through supporting a mapping from different data-models to the common SIOM model, and through facilities for doing schema-integration. *Presentation integration:* through separating functional objects and presentation objects and making functional objects easily accessible from interactive user-interface design tools. *Process-oriented integration:* by providing a basis for invocation of tools and notification of events, as a useful facility for a separate process management service.

Presentation integration is supported by a separation of user interaction components and service components. Process integration is supported by user-oriented process interaction engines and group-oriented process interworking engines. Data integration is supported by access to various database-services, possibly with an object-mapping layer. Control integration is supported by a software bus infrastructure for interaction between components.

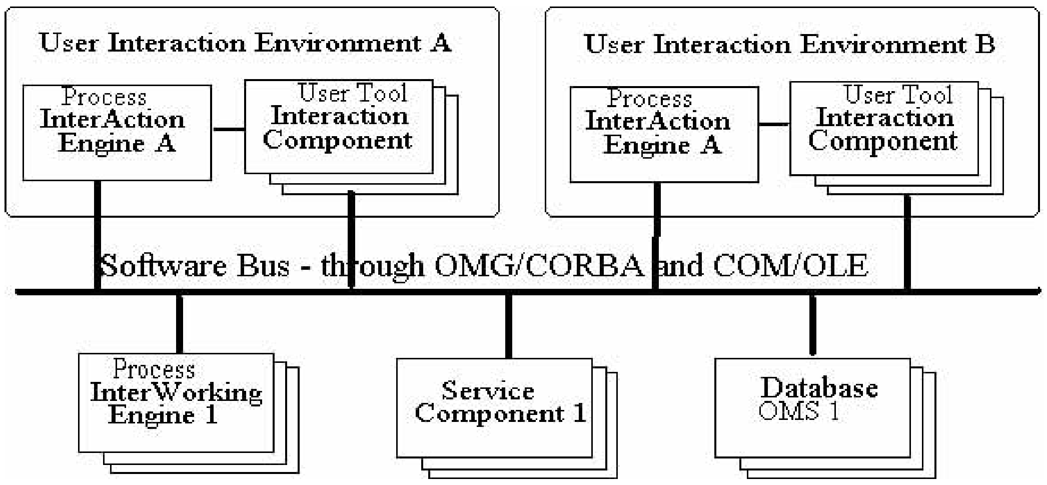

User Interaction Environment A
Process InterAction Engine A
User Tool Interaction Component
User Interaction Environment B
Process InterAction Engine A
User Tool Interaction Component
Software Bus - through OMG/CORBA and COM/OLE
Process InterWorking Engine 1
Service Component 1
Database OMS 1

Figure 4 SISIP - Reference Architecture

The SISIP reference architecture shows how a distributed persistent object space can be realized around a software bus architecture based on OMG/CORBA with COM/OLE interoperability and services for user interaction, database-support and persistence and work-processes.

4. SIOM - SISIP Object Model

The SISIP Object Model, SIOM, is fully object-oriented and has language-bindings to different object-oriented programming-languages, initially C++ and Smalltalk.
It is fully object-oriented, as defined in [Dittr86], as a merge of structurally and behaviorally object-oriented models. A fully object-oriented model can be viewed as a merge of the traditional data-modeling approach and object-oriented technology.

SIOM is inspired from our earlier work on object models: SOOM, Semantic Object-Oriented Model in Smalltalk-80 [Ber86,Ber88] and COOM [Ber93].
SIOM is based on behavioral modeling from the Object Management Group (OMG) Object Model, [OMG91], and structural modeling from the ODMG'93 object model and the EXPRESS information modeling language, with some extensions.

The following are some of the principles for SIOM:

- The representation of functionality and data of heterogeneous systems and databases as encapsulated objects
- All interaction happens through messages sent to encapsulated objects
- A set of operations "belonging" together can express certain semantics, i.e. attributes, relationships
- The use of three languages: SIODL - SIOML - SIOQL
- The separation between interface, implementation and extent
- The representation of run-time information about interfaces, implementations and extents

4.1 Objects and languages

The basic concept in SIOM is the encapsulation of data and functionality in objects that have an interface provided as a set of operations. The encapsulation gives a possibility for hiding underlying heterogeneous implementations. All interaction happens through messages sent to encapsulated objects. A set of operations "belonging" together can express certain semantics, i.e. attributes, relationships. Short-hands for specifying such common sets of messages are provided by SIODL, the Object Definition Language.

Manipulations of objects are done in SIOML, the Object Manipulation Language, which is a mapping of SIOM-concepts to object-oriented programming languages with C++ or Smalltalk as initial options. Objects might be created and retrieved in different ways. Retrieval of objects is done through SIOQL, the Object Query Language. A predicate is applied to the extent of an interface or an implementation, and will return the subset of objects that are satisfying the predicate. SIOQL is based on the ODMG'93 OQL query language.

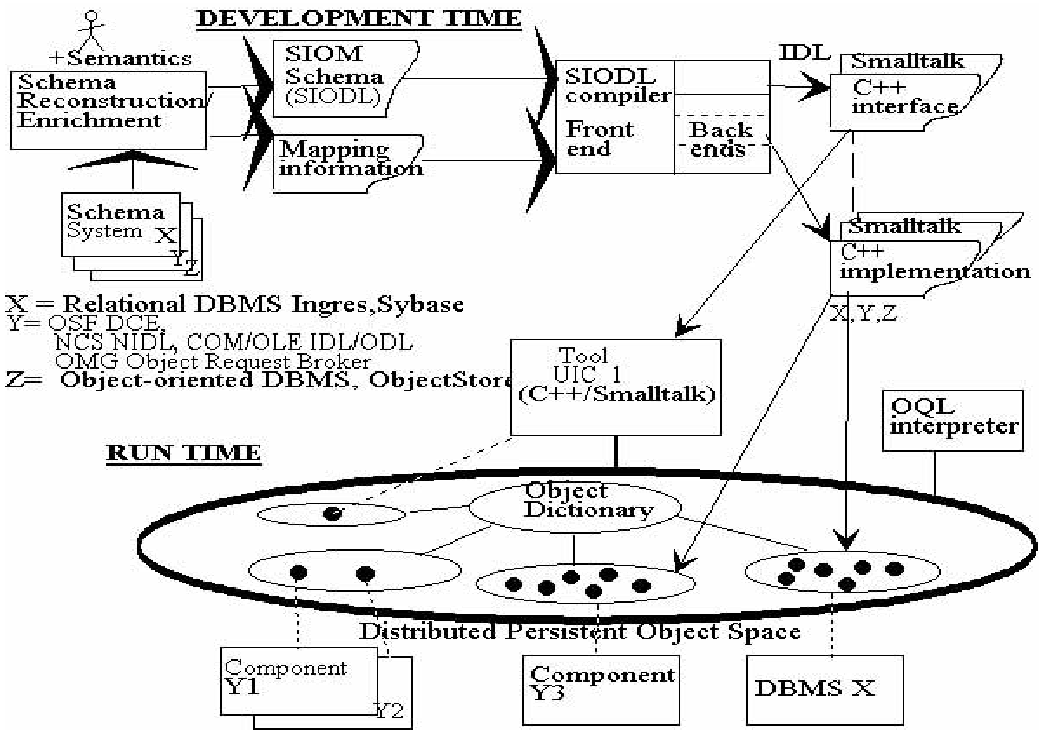

DEVELOPMENT TIME
+Semantics
Schema Reconstruction/ Enrichment
Schema System X Y Z
SIOM Schema (SIODL)
Mapping information
SIODL compiler
Front end
Back ends
IDL
Smalltalk C++ interface
Smalltalk C++ implementation
X,Y,Z
X = Relational DBMS Ingres,Sybase
Y= OSF DCE, NCS NIDL, COM/OLE IDL/ODL OMG Object Request Broker
Z= Object-oriented DBMS, ObjectStore
Tool UIC 1 (C++/Smalltalk)
OQL interpreter
RUN TIME
Object Dictionary
Distributed Persistent Object Space
Component Y1
Y2
Component Y3
DBMS X

Figure 5 SISIP at Development Time and Run Time

4.1.1 Separation between interface, implementation and extent

Most current object-oriented programming languages, like C++ and Smalltalk, use the class-concept for specifying interface, implementation, as a factory for creating objects and as a manager for all instances of a class (Smalltalk). In SIOM we make a distinction between these four concepts by a separation of interface, implementation, extent and object-factory services.

The notion of generalized and specialized interfaces is similar to supertypes and subtypes, as described in [Blair91]. Specialization is based on conformance for pluggability, - a specialized interface can be used wherever the generalized interface might be used.

The separation between interface and implementation allows for multiple different implementations for one interface. These implementations might be using different underlying systems which have different ways of identifying objects.

An interface-extent includes all objects which provides that interface. An implementation-extent includes all objects which have that implementation, and thus also the same interface. An extent is represented as a set-object.

4.2 SIODL - SISIP Object Definition Language

The basic principle behind SIOM is to facilitate for systems integration through encapsulated objects, using SIODL for defining the interface of objects. The extent of an interface is all objects that support this interface. This includes the extent of all different direct implementations and the extent of all sub-interfaces, or specialized interfaces, for this interface.

SIODL is aimed at the description of the interface of objects, as semantically complete as possible, and without any reference to implementations. SIODL merges concepts from the distributed system world, where interfaces typically are described as a set of operations through an Interface Definition Language, and concepts from the ER-oriented database-world where entities, attributes and relations are described through a Data Definition Language.

SIODL will lead to SIOML-based class-definitions where attributes are mapped to put- and get-operations, and relations are mapped to relation-management objects with a corresponding set of operations.

The separation between interface and implementation is an important part of the SIOM-support for systems integration and mapping to heterogeneous systems. This facilitates for the use of interfaces with multiple implementations. For each interface there might exist more than one implementation.

The SIOM object model is based on two layers. SIOM_B behavioral model for control integration and SIOM_S structural model for data integration. SIOM_S is realized as a layer above SIOM_B.

4.2.1 SIOM_B for Control Integration

The motivation for SIOM_B is to meet the requirements for request-oriented and notification-oriented control integration. A particular goal is to provide suitable support for these requirements through one mechanism. A basis for request-oriented control integration is adopted from the interface-definition-language OMG CORBA IDL. A basis for notification-oriented control integration is adopted from the HP Softbench/Field enhanced with a notation for the explicit specification of notifications in a schema.

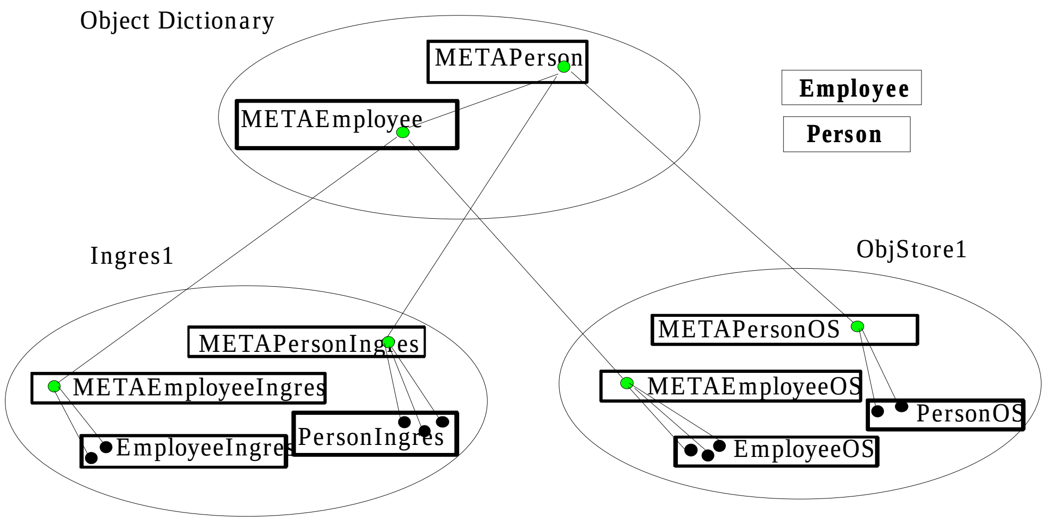


Figure 6 Interface Managers and Implementation Managers with objects

These mechanisms are integrated into a new object-management framework that separates between object interface managers and object implementation managers. The predefined functionality of objects and object managers is enhanced from functionality provided in the REBOOT datamodel [Reboot91].

```
SCHEMA PersonSchema {
   import Standards;
   import Basics;
   Notification  Changed
   {};
   Notification  ChangedAge:  Changed
   {int age;
   };

   INTERFACE Person {
  // META-operations are for manager objects
     META Person create(in String  name, in int  age);
     META int averageAge();
     META int nrOfPersons();
     String name();
     void name(in String newName);
     int age();
     void age(in int newAge) Raises(illegalAge) Notifies(ChangedAge);
     int personNumber();
     int personNumber(in int newPersonNumber);
   };  /* ObjInterface Person */

INTERFACE Employee: Person {
   META Employee create(in String name, in int age, in int salary);
   META int averageSalary();
   int salary();
   void salary(in int newSalary) Raises(illegalSalary)
             Notifies(ChangedSalary);
  };  /* ObjInterface Employee */

} /* SchemaPerson */
```

Notifications are introduced in order to support the requirements for notification-oriented control integration. Notifications is a mechanism for providing asynchronous indirect interaction among objects. The mechanism provides a facility for objects to notify about events, and for objects to register their interest for being notified about events. The specification of potential notification events, is supported by SIODL, because notifications generated from an object is an important part of the external knowledge about an object. SIODL provides a syntax for defining notifications with value attributes in a directed graph, and to specify which notifications might be generated after which operations from objects supporting a certain interface.

4.2.2 SIOM_S for Data Integration

SIOM_S is a structural modeling layer above SIOM_B that introduces extensions to the SIOM model for abstract attributes and relations, as a foundation for the support of data integration. This is also used as a basis for the definition of the Object Query Language, SIOQL. Abstract Attributes, Relationships and OQL provide support for single-model data integration.

In some kinds of applications it is natural to use the concepts of entities, relations and abstract attributes as basic modeling constructs. The group of applications this is valid for is typical database-oriented applications, where the goal is to update and retrieve information from the model itself. The focus is on the structure of the model, not on the behavior and interaction between the various objects. Our work with the EXPRESS information modeling language in various application areas, such as CIME, GIS and E&P, has shown that this is a useful approach for data modeling, but lacks the necessary facilities for object-oriented modeling, [Berr94]. Our approach is to use an EXPRESS-compatible syntax that is extended with support for operations.

```
ENTITY entityx SUBTYPE OF (super1, super2)
{/* Exception-events */
 /* Notification-events */
 /* Operations */
interfacez  op1(IN int x1, IN string s1)
        RAISES(Exc1)  NOTIFIES (Not1);
/* Attributes */
att2  : INT;
/* Unique attribute-combination */
UNIQUE  att2;
department : OPTIONAL Department;
};
```

Support for structural modeling should not violate the encapsulation principles of a behavior-oriented model. An important goal is to still keep the possibility for having different implementations of an interface at the same time. This is useful if a relation or an attribute is represented differently in different underlying systems. Therefore our approach for SIOM_S is to create a structural layer above SIOM_B, which uses EXPRESS-based constructs for modeling structural aspects. This can be mapped down to the behavioral layer based on OMG CORBA. In order to support object-oriented structural modeling we also aligns this with concepts from the ODMG OODB-standard proposal. In particular we adopt concepts from the ODMG OQL query language.
The relationship-modeling in both EXPRESS and ODMG is embedded in the Entity/Interface definitions, as opposed to being separate constructs as in our previous approach in COOM. To handle some of the inconsistencies between EXPRESS and object-oriented modeling we use some of the concepts discussed in [PISA94].

4.3 SIOML - the use of object oriented languages

SIOM is realized by mappings of the SIODL-specifications to class-definitions in an object-oriented programming language. The implementation-code is realized in an existing programming language. Each SISIP SIODL-schema will have a concrete language representation in different object-oriented languages, C++ and Smalltalk are the initial choices. The implementation-part of a class will be dependent on the characteristics of the underlying heterogeneous systems. This can partly be automatically generated, based on the mapping-process from local data-models/schemas to the SISIP Object Model.

The SIOML - Object Manipulation Language, is realized by mapping SIODL interfaces to constructs in existing object-oriented programming languages such as C++ and Smalltalk, by the use of language specific constructs for the actual implementation. The operations supported by all objects fall into the following two groups: Copy and delete operations and Information operations. The functionality provided by manager objects fall into the following three groups: Implementation- and subtype- management operations, Object-creation operations, and Object-retrieval operations.

4.4   SIOQL - Object Query Language

In order to efficiently deal with a large number of objects, there is a need for a powerful query-language. It is necessary to be able to select and retrieve existing objects, based on predicates on their characteristics. The initial basis for such selection is the extents of interfaces and implementations. A query on an extent will return the set of objects that conforms to the predicate in the query.

The Object Query Language (SIOQL) gives possibilities for selecting subsets of the existing collections of objects, based on predicates on their attribute-values and relationships. The goal of the query-facility is to transform a query formulated in SIOQL to a (set of) efficient queries on the underlying systems.

The first version of SIOQL has been based on the REBOOT Associative Query Language, the next version will be more conform with the ODMG query language, OQL, and the OMG/CORBA query service.

5. SIMOD - Based on OOram role modeling

The SISIP framework supports a new way of system building, by greater emphasis on reusing existing system-components. System building is essentially the process of gluing together pre-existing parts to fit new requirements. It is of major importance that objects are able to play different roles in different settings. A role is characterized by an interface which an object playing that role has to support.

We have created an object oriented development methodology that supports this way of building systems, OORASS - Object Oriented Role Analysis, Synthesis and Structuring, [Reen92], this has lately been developed further to the OOram method [Reen95]. The OOram methodology and tools provide a comprehensive environment for object oriented analysis, design and implementation in a wide range of application domains such as enterprise modeling, organization development, information management systems, user interfaces, real time systems, and telecommunication systems. Because of the separation of role modeling, interface specification and implementation, it is particularly suited for the modeling of distributed object systems.

The separation between roles, interfaces (types) and implementation (classes) has been a guideline in the development of OOram. The roles gives a description of an object in a given context. The interfaces of an object is given by the roles it play with respect to other objects. The implementation of an object can be done in different ways, and objects with the same interfaces might have different implementations. The OOram separation of why, what and how, provides a good basis for the analysis and design of distributed object systems. In the following we will present how OOram can be used for applications to be realized in an OMG/CORBA or COM/OLE environment.

SIMOD is an architecture-oriented system development method according to the SISIP architecture. It focuses on four different models: Process Model, User Interaction Model, Tool/Service Model and Data Object Model, corresponding to the four integration areas in SISIP.

SIMOD focuses on the development of OOram role models in order to create the following four object models:

- SIMOD Business/Work Process Object Model
- SIMOD User Interaction Object Model
- SIMOD Service/Component Object Model
- SIMOD Business/Data Object Model

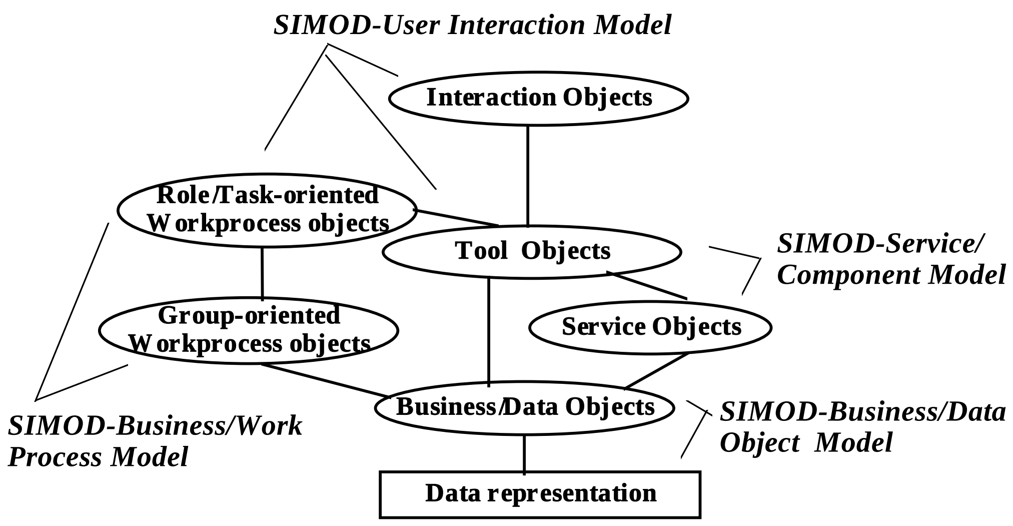

SIMOD-User Interaction Model
Interaction Objects
Role/Task-oriented Workprocess objects
Tool Objects
SIMOD-Service/ Component Model
Group-oriented Workprocess objects
Service Objects
Business/Data Objects
SIMOD-Business/Work Process Model
SIMOD-Business/Data Object Model
Data representation

Figure 7 SIMOD Four Sub-models

The process object model focuses both on general business process modeling, and on work process modeling. Techniques from BPR, Task-analysis and work-flow is used as part of the method.

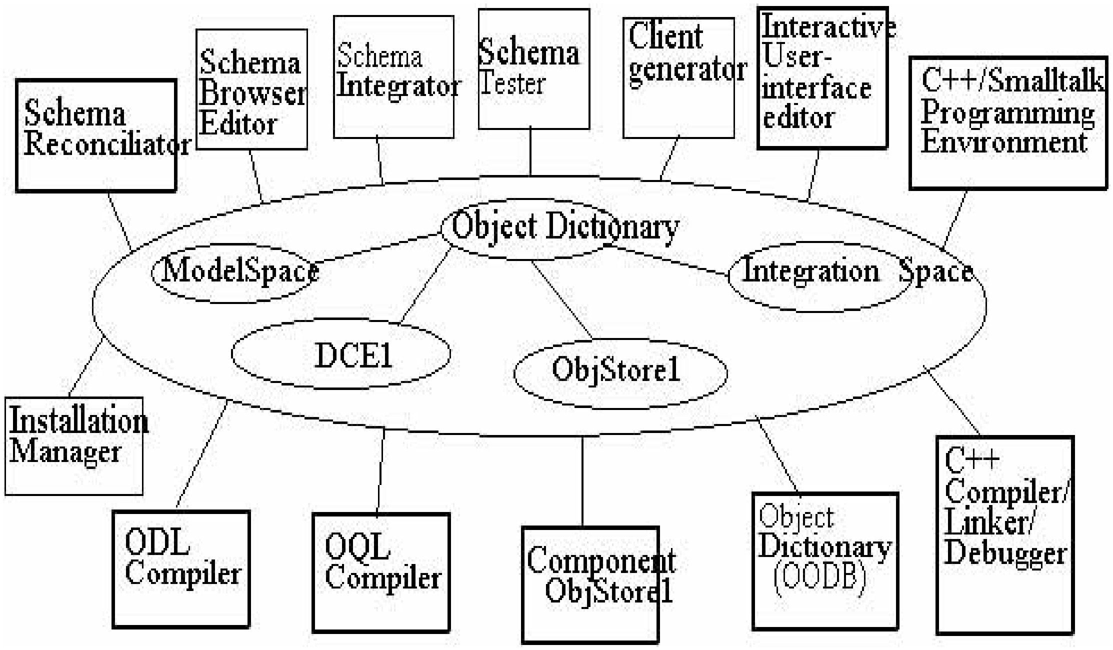
The user interaction model focuses on the specification of the user interface functionality and the dialogue for each tool. This is accompanied by a user interface prototype. Techniques from the Human-Computer-Interaction area is used as part of the method.

The service/component object model focuses on the modeling of how the tool functionality is realized by the use of new and existing system components and services. This is the main model area directly supported by the basic OOram methodology. In this area there is particular support for techniques for component integration for the integration of existing systems and applications.

The business/data object model focuses on the modeling of persistent objects and the relationships and constraints among these. Techniques from data modeling (Express, NIAM, ERA) and business object modeling (OMG BOMSIG, [Sims94]) is used as part of the method.

6. SIDE - SISIP Development Environment

We are specifying a development environment which supports this methodology, SIDE - SISIP Development Environment, with special support for the schema-integration process. SIDE is an extension to the CODEDISC environment which in particular focused on the development of client-programs in a component-based software bus environment [Berr92].



Combined with modern interactive tools for user interface building, this will be a powerful development environment for system integrators.

7. Specializations of SISIP

The SISIP approach is being applied in various domains.

The GEOOSIP architecture [Berr94] is applying the SISIP architecture to the domain of Geoscience (Exploration and Production). The GEOOSIP approach is validated through work done in the KEEP and ObjectSIP projects. In KEEP the GEOOSIP approach is be used for the existing TerraMod 3D Geological modeling system developed at SINTEF and Norsk Hydro.

GISSIP [Berr95] is focusing on an integrated environment for Geographical Information Systems. on a realization of the Open Geodata Interoperability Specification (OGIS) Geodata Model. The control integration part is based on an object-oriented client-server architecture for cooperative processing, based on the CORBA-technology from the Object Management Group.

In the area of CIME [Ber94b] the use of STEP/Express is particularly important. The integration of EXPRESS with object-oriented extensions in SIOM is therefore potentially quite useful here. We are exploring this approach in the context of some of our ESPRIT CIME projects, PRODEX, INTERROB and MARITIME.

8. Conclusion and future work

The SISIP approach extends from earlier work on the COOP architecture and the COOM object model. The differences is that the COOP architecture was based around an object oriented database and SISIP is based around an OMG/CORBA compliant object request broker with object services, ODMG object-oriented database integration and COM/OLE interoperability. The COOM object model was based on the combination of concepts from OSF/DCE and Entity-relationship modeling, with relations as a separate construct, and SIOM is based on CORBA IDL, ODMG ODL and the EXPRESS information modeling language. The SIMOD methodology is based on the OORAM method, while the COOP COORASS methodology was based on the earlier OORASS method. The SIDE development environments extends on the COOP CODE environment, with a SISIP compliant architecture.

The contribution of the SISIP Integration framework is to provide a unified approach to integration combining data-integration, control-integration, process integration and presentation-integration. It has been successful to extend our earlier work on software bus technologies, [Berr92], to also cope with more fine-granuled objects.

The current work is focusing on finalizing the mappings to and use of the new technologies introduced in SISIP compared to COOP. Some other features we are currently working on for SISIP are Relational database mapping and integration, Multi-model data integration, Express-OO standard proposal, SIOQL Query processing, OMG/CORBA and COM/OLE interoperability, SIMOD/OOram for Distributed Objects and OODBMS.

A number of areas are on our work-topic list, such as: Transaction Management, Dynamics and evolution, Schema reconstruction and mapping, Schema integration and multi-model integration, Query Processing, Security Support, Integration with a trading language, Work Process Integration and Semantics of behavior and object interaction.

Future work will be on experimenting with the integration framework for different underlying heterogeneous systems, to get further experiences from practical use of the framework. We will also look closer into transaction-mechanisms, security-issues and the trade-offs between different degrees of autonomy for underlying systems. Different strategies for distributed query-processing will also be looked into. We will define some extensions to SIODL which will enhance it to play the role as an object-integration-language for doing semantic integration.

Bibliography

[ABM+90] T. Lougenia Andersson, Arne J. Berre, Moira Mallison, Harry Porter, and Bruce Schreider. “The HyperModel Benchmark”, Proc. of EDBT’90, Extending Database Technology, Springer LNCS no 416, pages 317-331, March 1990, Venice.

[Berr95b] Arne-Jørgen Berre, Bjørn Hjelle, David Skogan
"GISIP - An Object-Oriented Software Integration Platform for Geographical Information Systems" Proceedings of SCANGIS'95 - Scandinavian Conference on Geographical Information Systems, Trondheim, June 1995.

[Ber94a] Arne-Jørgen Berre and Magnus Rygh “GEOOSIP - An Object-Oriented Integration Framework for GeoScience Applications” New Trends in Geoscience computing, NPF, November 1994, Stavanger, Norway

[Ber94b] Arne-Jørgen Berre and Frode Høgberg, Object-Oriented Design and Implementation of an EXPRESS-based Product Model Database - On Object-Oriented Extensions to EXPRESS, EUG'94, EXPRESS User Group Conference, October 12-13 1994, Greenville, South Carolina, USA

[Berr93] Arne-Jørgen Berre "An Object-Oriented Framework for Systems Integration and Interoperability" PhD-thesis, University of Trondheim, Norwegian Institute of Technology, 1993, 370 pages

[Berr92] Arne-Jørgen Berre
"COOP - An Object-Oriented Framework for Systems Integration"
Proceedings of second international conference on Systems Integration, June 1992, Morristown, N.J. , USA

[Berr92a] Arne-Jorgen Berre "Experiences from Systems Integration through an Object-Oriented Software Bus" In Proc. of TOOLS-Europe'92, Tool2-7, Prentice Hall, pages 33-46, April 1992, Dortmund, Germany

[Berr92d] Arne J. Berre. COOM - An Object Model for a Systems Integration Framework. In Proc. of the International Workshop on Distributed Object Management, August 1992, Edmonton, Canada, Morgan Kaufmann Publishers, 1993

[Ber91] Arne J. Berre The HyperModel Benchmark for evaluating Object-Oriented Databases In Object-Oriented Databases - Systems and Applications, pages 75-91, Prentice Hall, 1991

[Ber88] Arne J. Berre, SOOM and Tornado - Experience with Database-support for Object-Oriented Applications, In Proc. of 2nd International Workshop on Object-Oriented Database Systems - Bad Munster, Germany, September 1988

[Ber86] Arne J. Berre Sharing of Objects in an Object-Oriented Language. In Proc. of 1st International Workshop on Object-Oriented Database Systems, Asilomar, USA, September 1986

[Blair91] Gordon Blair et. al. "Object-oriented Languages, Systems and Applications" Pitman, 1991, 378 p.

[Brod92] Michael Brodie, Stefano Ceri "On Intelligent and Cooperative Information Systems" International Journal of Intelligent and Cooperative Information Systems 1(2):1-35, September 1992

[Dittr86] Klaus R. Dittrich "Object-oriented database systems: The notion and the issues." Technical Report, Forschungszentrum Informatik (FZI) and der Universität Karlsruhe, 1986. Preface in 2nd International OODBMS Workshop 1988.

[ECMA90] European Computer Manufacturer's Association.
"A Reference Model for Frameworks of Computer Assisted Software Engineering Environments.", Technical Report, ECMA/TR55, December 1990.

[Nils90] E.G. Nilsson, E. Nordhagen, G. Oftedal
"Aspects of Systems Integration", Proceedings of first international conference on Systems Integration, April 1990, Morristown, N.J. , USA

[OMG91] Object Management Group, OMG, 492 Old Connecticut Path, Framingham, MA 01701, USA.
“OMG CORBA Common Object Request Broker Architecture - Specification”, OMG Document Number 91.12.1, Revision 1.1., September 1991

[PISA94] Günther Staub, editor.
“PISA Information Modelling Language: EXPRESS-C”, Working document of ISO TC 184/SC4 WG5, N202. October 7, 1994.

[Reboot91] Erik Odberg and Uwe Hohenstein
“REBOOT Data Model and Database Interface specification”, REBOOT Project Deliverable D1.2.B1, NTH and Siemens, Dec. 1991, 142 pages

[Reen92] T. Reenskaug et.al.
"OORASS - Seamless Support for creation and maintenance of Object Oriented Systems", Journal of Object-Oriented Programming, Vol. 5, No. 6, page 27-42, October 1992

[Reen95] T. Reenskaug et. al
"Working with Objects - The OOram way to software success"
Book in print - to be published, Addison-Wesley, summer 1995

[Sims94] Oliver Sims
"Business Objects - Delivering Cooperative Objects for Client-Server"
The IBM McGraw-Hill Series, 1994, 348 pages

[Thomas92] Ian Thomas and Brian Nejmeh
"Definitions of Tool Integration for Environments ",
IEEE Software, Vol. 9, No. 2, page 29-35, March 1992

[OOram95] Taskon
"OOram 3.0 User's Guide", March 1995

[Webster] Virginia Thatcher and Alexander McQueen
"The New Webster Encyclopedic Dictionary"
Consolidated Book Publishers, 1980, 972 pages