OMNI/03/01

Egil P.Andersen

Oslo

June 2001

Norsk Regnesentral
ANVENDT DATAFORSKNING

NR-notat/NR Note

Tittel/Title: SINAI - Part II: Tools and Technology
A UML-based Architectural Framework for
Evolutionary Information Systems

Dato/Date: 13.June
År/Year: 2001
Notat nr: OMNI/03/01
Note no:

Forfatter/Author: Egil P.Andersen

Sammendrag/Abstract:

SINAI is made to support the development of web-based distributed information systems that are *open*, *generic* and *extendable*; i.e., the kind of information these systems must be able to manage is not fully known at design-time. They evolve over time and must be designed for this. A SINAI based system will have many of the characteristics of a generic system, but without being generic itself. It will *not* predominantly consist of generic software components. Instead it will consist of dynamically composable units of information and dynamically configurable application components, and it will also allow for domain-specific specialisation, customisation and optimisation.

This report, *"SINAI Tools and Technology"*, describes a prototype made to get feedback for further development, and to get some feeling for the practical usability of SINAI. Notice that when reading this report you are expected to have read the other SINAI report, *"SINAI Concepts and Principles"*, OMNI/02/01. This latter report first presents the motivation and background for SINAI. Then it describes the novel aspects of SINAI from a conceptual viewpoint; i.e., independent of the specific technology used to realise the "proof of concept" prototype described herein. The novel aspects of SINAI primarily concerns using role modelling techniques, from object-oriented analysis/design, as an architectural principle when designing and also implementing enterprise, distributed, layered, often web-based, information systems.

SINAI is funded by "utviklingsfondet" at the Norwegian Computing Center.

Emneord/Keywords: Evolutionary, distributed information systems, internet, UML, XML, software components, COM

Tilgjengelighet/Availability: Open

Norsk Regnesentral / Norwegian Computing Center
Gaustadalléen 23, Postboks 114 Blindern, 0314 Oslo, Norway
Telefon 22 85 25 00, telefax 22 69 76 60

Contents

1. Project Summary - A SINAI Prototype for "Proof of Concept"

Our main motivation for SINAI was an assumption or hypothesis that object-oriented analysis/design could prove useful also from an architectural and implementation point of view when developing enterprise information systems. SINAI particularly focuses on the problem of how to realise evolutionary information systems, e.g. like electronic patient record systems where one cannot foresee at design-time all the different kinds of information that the system should be able to manage some time in the future. Hence, more specifically, our hypothesis for SINAI is that the principles of role modelling [32-67, and see the other SINAI report] can make the realisation of evolutionary information systems possible without undue effort or risk of failure, and this while avoiding many of the known problems and disadvantages of generic systems; i.e., systems that predominantly consist of generic software components. This is the background for the *SINAI Information Models (SIM)* and *SINAI Application Models (SAM)*, and SINAI standardised component interfaces (COM/IDL) and relational database stored procedure interfaces for accessing and working with objects in these models, as explained in-depth in the other SINAI report, *"SINAI - Concepts and Principles"* [1].

Developing SINAI systems by hand is very time-consuming, but it is possible to automate parts of the development process by generating parts of the code that is needed to adhere to SINAI conventions. Thus the other main task in the SINAI project, beside considering how to utilise role modelling, has been to implement a SINAI code generation prototype. The current prototype involves storing and generating Rational Rose models, generating IDL component interface specifications, generating Visual Basic components for information caches, and generating relational database schemas and stored procedure interfaces based on SIMs. This prototype will also be useful to enable an evaluation of the SINAI approach since it makes it easier to use SINAI on new projects without too much effort. The rest of this document will describe the SINAI prototype in further detail.

Notice that SINAI itself is *not* platform specific. Its principles can be applied equally well to e.g. J2EE and Microsoft COM or .Net. The current SINAI prototype is implemented on a Microsoft COM platform, however. Hence fluent knowledge of COM is required to make good use of it.

Notice also that the current prototype is just that, a prototype primarily for "proof of concept". Much more effort will be needed to turn it into a full-scale, possibly commercial CASE tool. This may never be a goal for SINAI, however. If an overall SINAI evaluation turns out positive then the SINAI principles of evolutionary enterprise information system development will be its most important contribution.

2. SINAI Code Generation and Standardised Architectures

2.1 SINAI Code Generation

A benefit from using SINAI models to generate parts of the system code, beside saving some coding, is also that this in itself will support a more standardised architecture; at least for a particular product line.

Of course, the usefulness of automatically generated code should not be overestimated. As will be explained in the other chapters of this document, in SINAI manual customisation and specialisation is possible at all levels and all layers of the architecture. For example, complex business rules usually require non-trivial manual intervention. Thus the main challenge is the organisation of predefined code versus automatically generated code versus case-specific manually made code, and this is where component technology provide a great benefit with its distinction between interfaces versus implementation, and the ability to change encapsulated components independently.

Information Models versus Application Models

As explained in the other SINAI report [1], SINAI makes an explicit distinction between *"information models"*[1] versus *"application models"*.

Information models define (stateful) objects residing on persistent storage (primarily databases), and their interfaces (i.e., SINAI COM IDL- and RDB stored procedure interfaces) contain functions for creating, removing, updating and retrieving objects (or "role instances" in SINAI). The data layer implementation of these functions handle information specific, but application independent, constraints.

Application models are service-oriented (often originating from Use Cases) in the sense that each model focuses on a particular set of related services (as a sub-application) that can be applied to particular information models. The application layer objects that implement application models are mostly *stateless objects* as required for scalability of large-scale information systems; see section 2.1.6 in [1].

SINAI Models and SINAI Databases

Figure 1 illustrates the principle of how SINAI code generation works. It is not as complex as it may seem at first. Its seeming complexity is due to the reflexive nature of SINAI and its code generation; e.g., much of the SINAI prototype can be generated from itself.

In SINAI databases will be the authoritative source of model information, not e.g. model files like .mdl files of Rose. There are several reasons for this. Model information in a unified database format will allow for modelling tool independence, i.e., Rose and Visio models can be interchanged. Model information can be made available to clients similar to other business information; i.e., authorised clients can have access to model information the same way they will have access to objects implementing these models, and model repositories can be distributed similar to how object instance databases are distributed. Databases are far better for maintaining information and data, versioning, back-up, etc, than piles of individual files.

To the left in figure 1 there is a set of SINAI models, initially defined in e.g. Rational Rose[2]. The four top left models are all SINAI Information Models (SIMs - illustrated in figure 2, 3, 4 and 5 below), while the bottom left model is a SINAI Application Model (SAM).

---

[1] The distinction in [5] between *"implementation models"* versus *"interface models"* only applies to SINAI information models, as explained in chapter 5 of [1].

[2] which is supported by the current SINAI prototype, but later on Visio2000 may well be a better alternative since it is both less expensive and much more powerful and flexible from a programming and development perspective.

The SIM meta-model, i.e., a model containing information about SIMs, is the "format" in which information about SIMs will be made available to clients. Figure 2 illustrates this model (and see chapter 5 in [1] for a detailed description of it). The SIM meta-model is used to implement a database, the "SIM meta-model database" in figure 1, which is used to store information about all SIMs (i.e., "all" for a particular SINAI server site). Hence information about SIMs will reside in SIM meta-model databases.

Consequently, the other 3 SIMs in figure 1, namely the SAM meta-model in figure 3 (which contains information about SAM models, similar to the SIM meta-model for SIMs), the RDBS meta-model in figure 4 (meta-information on relational database schemas) and any other SIM as e.g. the one in figure 5, will all be stored in a SIM meta-model database. Hence the SINAI prototype includes a service for storing a SIM Rose model into a SIM meta-model database (the "save" arrow in figure 1), and also a service for reading a SIM meta-model database and generating a SIM Rose model based on the information in the database (the "generate" arrow in figure 1).

The SAM meta-model in figure 3, which contains information about SAMs, is the "format" in which information about SAMs will be made available to clients; i.e., this similar to the use of the SIM meta-model. Thus the SAM meta-model is used to implement a corresponding SAM meta-model database, as illustrated in figure 1. Hence the SINAI prototype should include (but not implemented yet) a service for storing a SAM Rose model into a SAM meta-model database (the bottom "save" arrow in figure 1), and also a service for reading a SAM meta-model database and generating a SAM Rose model based on the information in the database (the bottom "generate" arrow in figure 1).

Similarly, the RDBS model in figure 4, which contains information about relational database schemas, is used to implement a corresponding database for storing information on database schemas.

Finally, every SIM that models persistent objects will give rise to its own corresponding database. Thus a database schema will be made (or generated by the SINAI prototype) for domain-specific models like the one in figure 5 below.

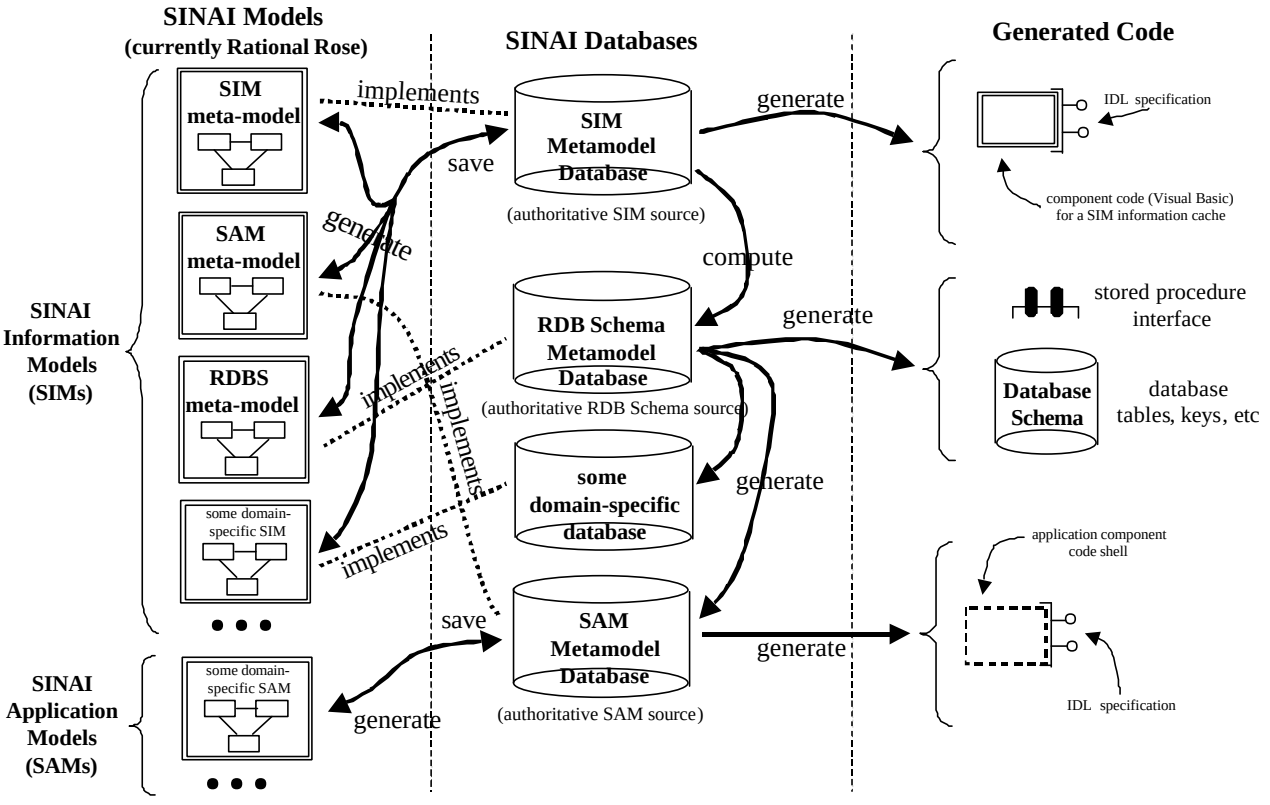

Figure 1. *SINAI code generation.*

Database, IDL and Component Generation

SINAI SIM and SAM model information can be used to generate at least the following kinds of code.

- IDL and component code (Visual Basic for the current prototype) can be generated to implement a cache for storing information on objects of a particular SIM. Such caches are particularly useful to "fat" clients for storing and working with information when disconnected from a server, but they can also be used when working with SIM objects on the server-side.
- Relational database schemas from SIMs.
  For reasons described in chapter 9, the database schema includes a set of stored procedures which together constitute an interface that encapsulates details of the how the tables of the database are defined and organised.
- SAMs, and application models in general, cannot be used to generate general-purpose application code, but SAMs can be used to generate IDL specifications for components that will implement corresponding application components; either on the client or server side. From IDL specifications it will be easy to get COM code shells in e.g. Visual Basic or Visual C++, thus avoiding some trivial development work.

As illustrated in figure 1, SIM information in the SIM meta-model database can be used to generate IDL and component code for SIM caches (the top "generate" arrow), and also relational database schemas. This latter generation is a two-step process, however. First SIM information in the SIM meta-model database is used to compute information about the relational database schema (based on user choices, different database schemas can be generated for the same SIM). This schema information is then stored in the RDB Schema meta-model database (the "compute" arrow). Later on information in the RDB Schema meta-model database can be used to generate the actual SQL script code[3] required to install the database schema on a particular database server (the mid "generate" arrows).

Correspondingly, SAM information in the SAM meta-model database can be used to generate IDL specifications and code shells for corresponding application components (the bottom "generate" arrow).

[3] or for MS SQL Server its DMO interfaces can be used to access the database programmatically with COM.

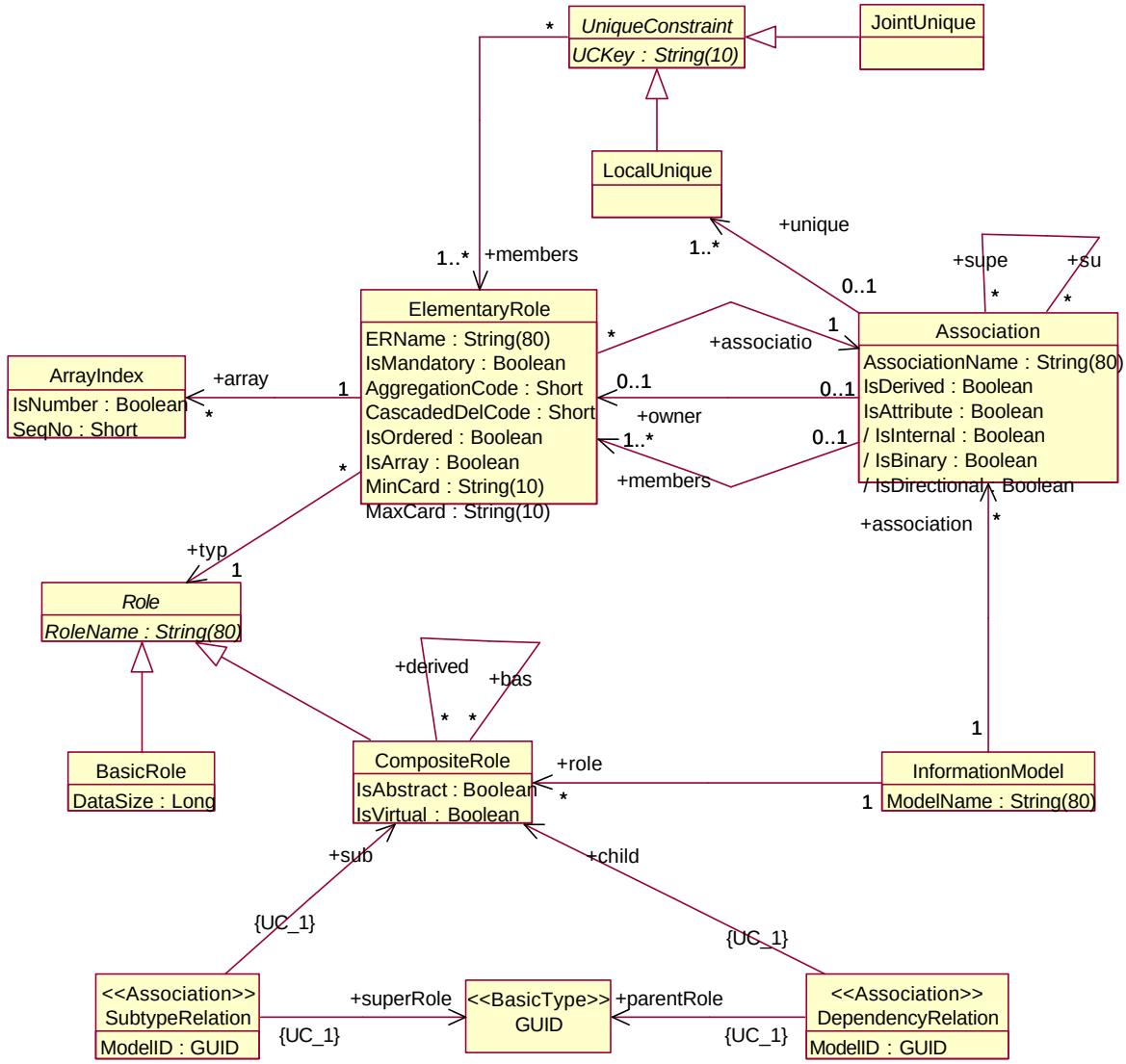

Figure 2. *Meta-model for SINAI Information Model (SIM Meta-Model).*

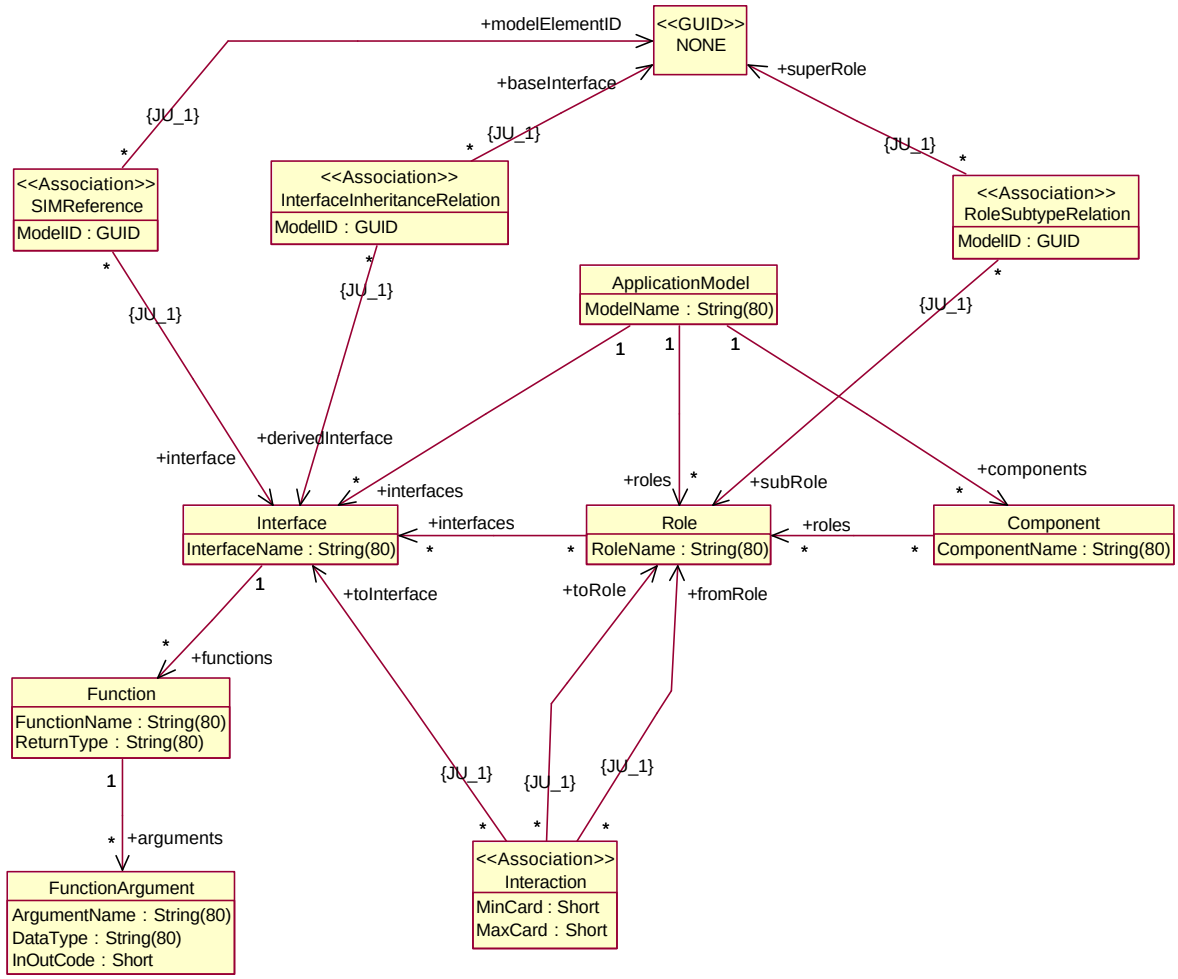

Figure 3. *Meta-model for SINAI Application Model (SAM Meta-Model).*

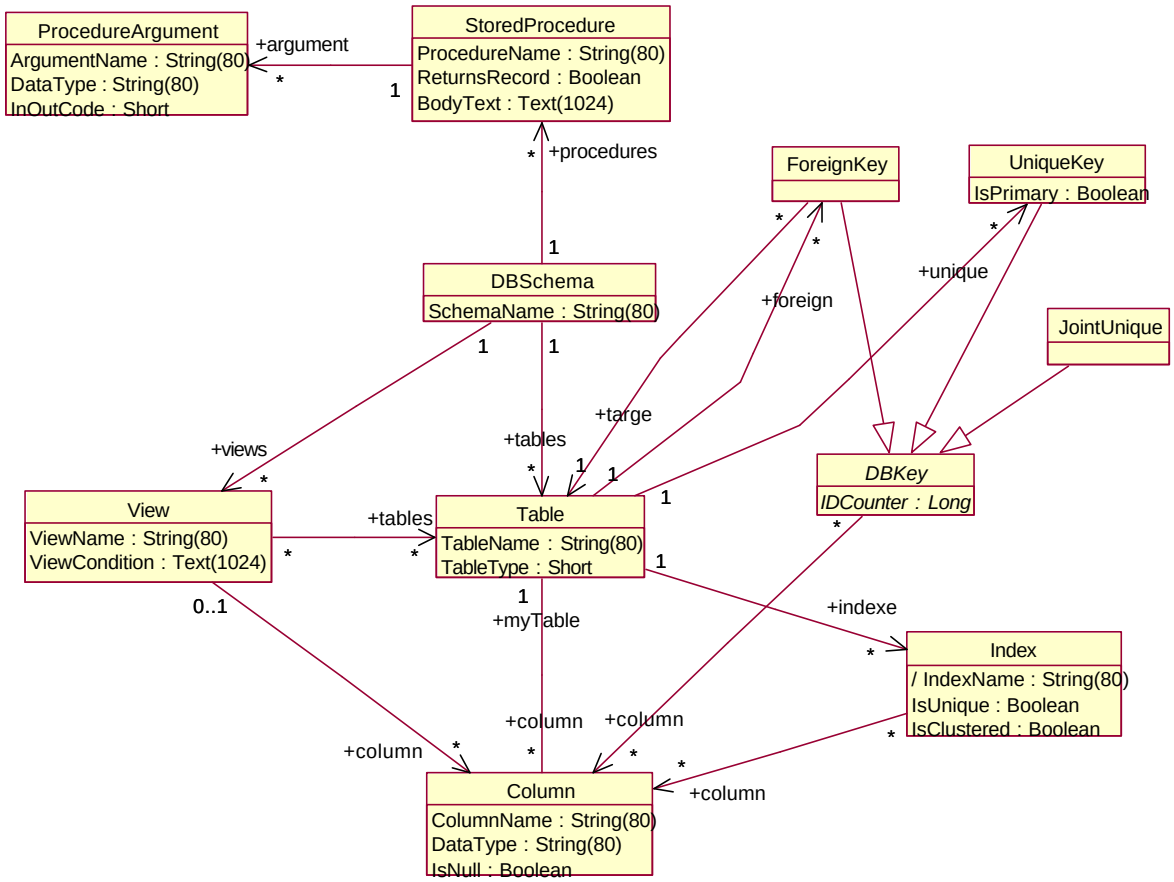

Figure 4. *Meta-model for Relational Database Schema Model (RDBS Meta-Model).*

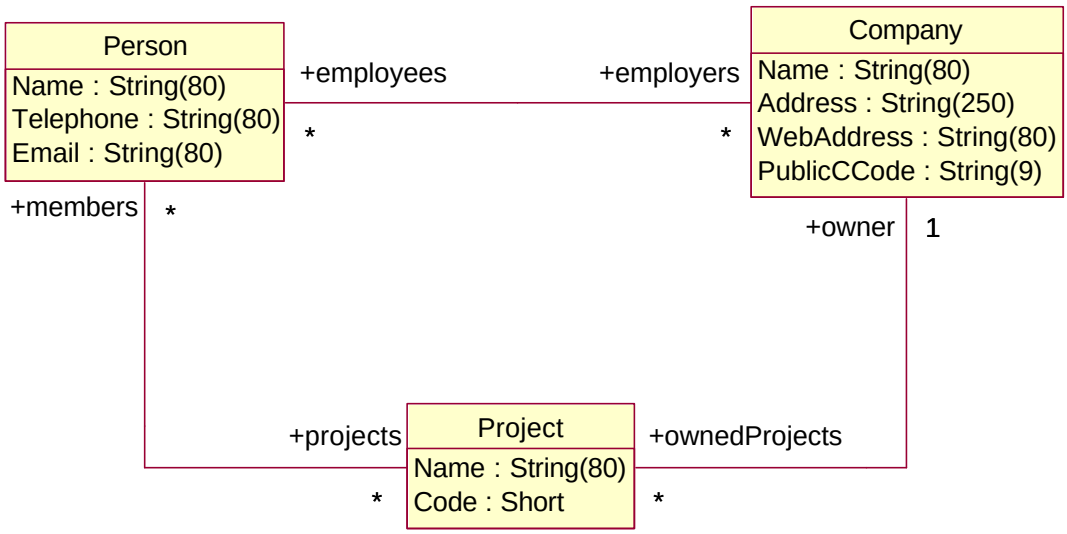

Figure 5. *Some domain-specific SIM.*

2.2 Some SINAI Architectural Styles

2.2.1 Check In/Check Out Style Information System

Figure 6 illustrates two different kinds of systems, or "architectural styles", that will be typical for SINAI based systems.

The first one, whose client components are illustrated top left, is a "check in/check out" style information system. In this system a client starts a session by requesting a, typically relatively large, set of information from the server. The information received by the client is then marked in the server/database as "checked out". This to be able to enforce some set of rules for how, or if at all, other clients can access this same information while being used by our client.

During his session the client read and write the received information, and he may well be disconnected from the server during the session. The information may also be saved to, and later loaded from, the local file system.

After finishing his work on the information, the client logs on to the server and sends the information, either all or at least those parts that have changed, back to the server. The server then handles the received information according to whatever server-side application components have been developed for this system, but eventually changes in the original information is stored in the appropriate databases.

SINAI Components

Figure 6 illustrates some components that may be involved in a "check in/check out" system like this. The client has a *cache* component that is used to store the information received from the server. This cache also offers to save its information to the local file system, or alternatively load information from the local file system.

Notice that when we use the term *"cache"* or *"SIM cache"* then we refer to a particular kind of software component, not caches in general, namely a component that will be able to store SIM information, and which offers SINAI standardised interfaces (IDL COM interfaces; see the other SINAI report [1]) for accessing this information.

When a client receives information from a SINAI compliant server then the information will always be formatted in XML according to a particular SIM. However, a SIM cache is SIM-dependent in the sense that it can only store information formatted according to a particular SIM. Also, its interfaces for accessing its information are also SIM-dependent interfaces. The SINAI prototype tool has services for generating a SIM cache based on a particular SIM; see section 2.2.3 below.

While SIM caches are useful to "fat" clients, they can also be used on the server-side. For example, a SIM cache (currently a set of ADO recordsets) is able to convert its content into SINAI XML (currently ADO XML). Thus when the server receives a client request for certain information, the information retrieved from the database may first be put into a server-side cache[4]. Afterwards the cache' service for converting itself into XML may be used to produce XML, which then comprise the server response back to the client (i.e., as an XML string return type of a SOAP function). When the client cache receives the information as XML it uses its function to convert SINAI XML into a SIM cache to fill itself with content.

Later, when returning to the server the updated information, the client cache correspondingly uses its cache-to-XML function to create the XML sent to the server. The server can then use a corresponding SIM cache object to receive the XML, create itself, possibly do some server-side processing, and finally store itself in the appropriate database.

Consequently, for information systems like this then a SIM cache with functionality as just described can be very useful to both the client- and server-side of the architecture.

[4] if SIM caches, as in the prototype, are based on OLE DB or ADO, then doing this is not a detour...

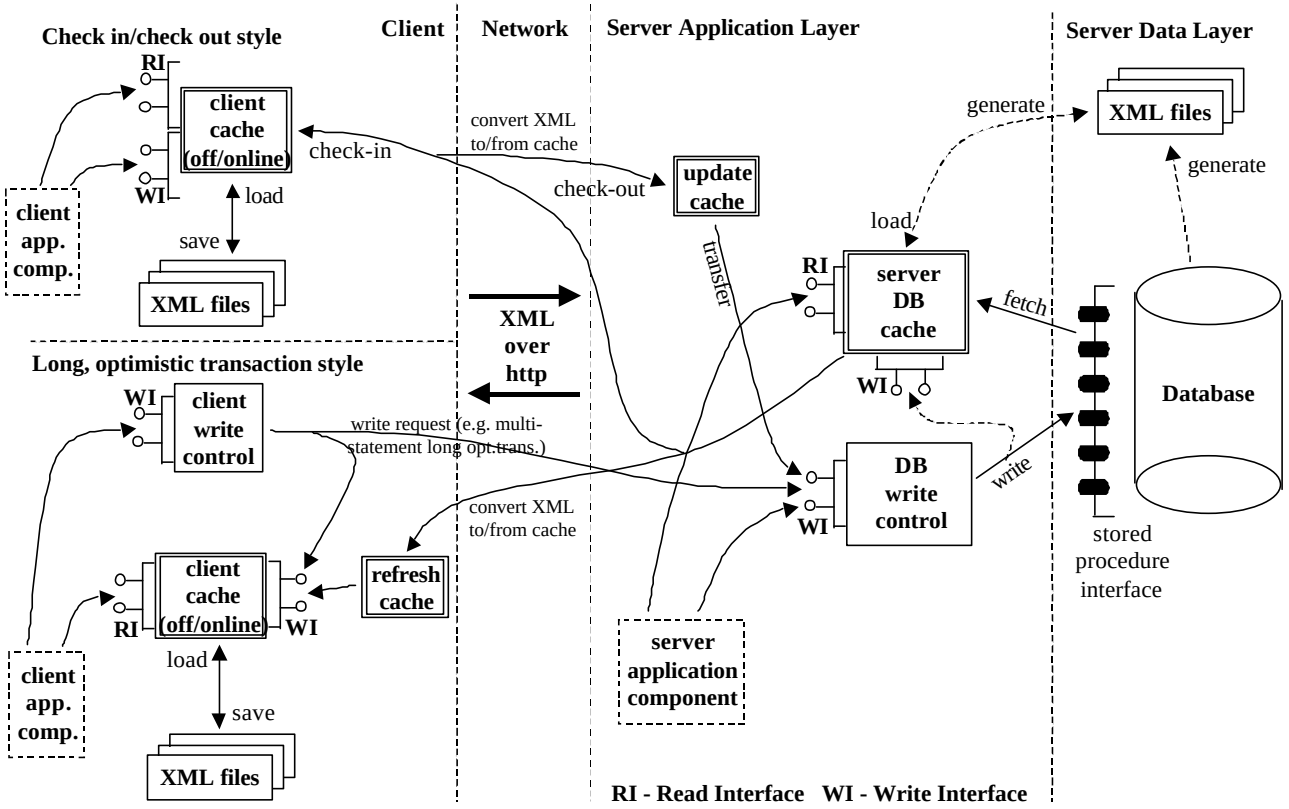

Figure 6. *Two example SINAI architectures with "fat" clients*
a) Check in/check out style information system
b) Long, optimistic transaction style information system

2.2.2 Long, Optimistic Transaction Style Information System

The other system in figure 6, whose client components are illustrated bottom left, is a system where no single client can, more or less, reserve certain information as in the "check in/check out" case. Instead many clients may attempt to access the same information at the same time, both for read and write, and transaction management and concurrency control techniques must be used to assure consistency in the data layer and among clients. Both the SHS EPR system and the DNV/BRIX system are examples of systems of this kind.

Transaction Management and Concurrency Control

This system may also have a (SINAI) cache on the client, similar to the "check in/check out" system, which may work both on/off-line, but in addition it must use a much more elaborate transaction handling scheme. It may have two levels of transactions; *long optimistic transactions* on the client-side, and *short pessimistic transactions* on the server-side. A client transaction consist of a sequence of operations on its cached objects, e.g. create object, delete object, update object, etc. No locks are held in any server-side databases during a client transaction. When the client attempts to commit then its transaction is sent to the server. The server then "plays" the transaction, i.e., executes in sequence each operation of the client transaction, and maintains database locks during this execution. Eventually, unless any errors or inconsistencies are detected, the server commits its transaction and notifies the client.

The optimistic concurrency control mechanism [15] used during client transactions can be as follows. Every (SIM) information object is equipped with a timestamp that records when the object was last updated (or created), and with a userID identifying the user that committed this change. When

an object is cached in the client its timestamp and userID are included. When a client transaction is sent to the server to update the objects involved, for each operation in the transaction the objects involved are validated with respect to their timestamp and userID as follows:

```
If database-object.timestamp = client-object.timestamp Or
   database-object.userID    = client-object.userID Then OK
                                                    Else Failure
```

That is, the operation fails if anyone other than the current user has made a change to the objects involved since they were sent to the client. A transaction operation that satisfies the validation check leads to a corresponding database update. If any operation within the client transaction fails then this leads to a failure, and immediate rollback, of the entire server transaction.

In general, for long transactions then a pessimistic concurrency control strategy can lead to long and unnecessary delays, and possibly time-consuming deadlocks. Optimistic concurrency control allows for more concurrency between different users, resulting in smooth work processes. Read-only access is never delayed, but for write access then conflicts may occur that are not actual conflicts. This depending on the granularity of the validation control. For example, if the validation is per object then there will be a concurrency conflict if different users change the state of the same object, despite that they may work on different unrelated properties of this object.

For an optimistic concurrency control strategy to be acceptable for long transactions the probability of concurrency conflicts must be sufficiently low. Noticeably, a validation failure is severe and not solved by a “simple” abort and retry of the transaction by a transaction manager. Instead the user is required to redo the read phase, and this may have to be done manually (depending on the support for handling conflicts). Thus designing good transaction management and concurrency control strategies is very much a non-trivial task for most enterprise information systems.

Client Refresh

The main differences between systems of the latter kind versus "check in/check out" systems concern transaction handling and concurrency control, but there is also an issue of client refresh. If it is possible that another client updates information currently cached by our client, then we may have to implement a cache refresh mechanisms that updates our cache with information that has changed on the server since the last refresh. As illustrated in figure 6, a SIM cache can also be used to keep the refresh information since there is no principal difference between SIM information received after the original client request, versus refresh information that the client requests (or the server "pushes") automatically at certain intervals.

2.2.3 SIM Caches based on SINAI Information Models (SIMs)

Chapter 8 below describes how the SINAI prototype can be used to generate SIM caches implemented in Visual Basic, and using ADO (Active Data Objects) recordsets to store cached SIM information. The following is an overview (see also chapter 6 in the other SINAI report [1]) of the COM interfaces and functions supported by every SIM cache.

Notice: Terms like *"composite role"*, *"bi-association"*, *"dir-association"*, and others, are defined in chapter 6 of the other SINAI report [1].

interface ICacheManager

This interface is common to every SIM cache.

`State``([out, retval] short*);`
Returns the current state of the cache

`ConnectToSource``([in] BSTR inSource);`

Set the SINAI source to which the cache information belongs

```
HasSourceConnection([out, retval] VARIANT_BOOL*);
```
The cache has a current SINAI source connection

```
CreateNewCache();
```
Creates a new and empty cache

```
CreateXMLFromCache([out, retval] BSTR*);
```
Creates SINAI XML for the current cache

```
CreateCacheFromXML([in] BSTR inXML);
```
Creates a cache based on the provided SINAI XML

```
SaveToSource();
```
The cache is saved to its SINAI source (fails if it has no source connection)

```
SaveToFile([in] BSTR inFileName,
           [in] VARIANT_BOOL inOverwriteFile,
           [out, retval] VARIANT_BOOL*);
```
The cache is saved to file as SINAI XML

```
LoadFromFile([in] BSTR inFileName,
             [out, retval] VARIANT_BOOL*);
```
The cache is loaded from a SINAI XML file

```
InsertRecordset([in] BSTR inRSName,
                [in] _Recordset* inRecordset,
                [in] short  inMode);
```
The specified recordset is inserted into the cache. If no such recordset currently exist, then this will become the new cache recordset. Otherwise information in the argument recordset will be inserted into the corresponding cache recordset if the information does not already exist in the cache (duplicates are removed).

```
VerifyCache([in] VARIANT_BOOL inRemoveInconsistency,
               [out, retval] VARIANT_BOOL*);
```
Not currently implemented.

```
ResetObjectStates();
```
Every recordset row with _SCode < 10 gets a new _SCode = 0

```
RemoveDeletedObjects();
```
Every recordset row with _SCode >= 10 is removed from the recordset.

interface ISINAIInformationModel

This interface is common to every SIM cache, but its functions vary depending on the SIMs composite roles and bi-associations.

```
CastAsCacheManager([out, retval] ICacheManager**);
```
Returns the same object as an ICacheManager

```
Get<CRname>It([out, retval] I<CRname>**);
```
Returns an iterator for the specified composite role <CRname>

```
Get<nBiAssName>It([out, retval] I<nBiAssName>**);
```
Returns an iterator for the specified non-binary bi-association <nBiAssName>

```
BeginTransaction([out, retval] BSTR* transID);
```
Starts a new transaction (the transID is not currently in use)

```
CommitTransaction([in] BSTR transID, [out, retval] VARIANT_BOOL*);
```
Commits a current transaction (the transID is not currently in use)

```
RollbackTransaction([in] BSTR transID, [out, retval] VARIANT_BOOL*);
```

Rollback on the current transaction (the transID is not currently in use)

```
GetNewGUID([out, retval] BSTR*);
```
Returns a new GUID made by the MS GUID generator

interface ICommonIterator

This interface is common to every composite role, bi-association and dir-association.

```
State([out, retval] short*);
```
Returns the current state of the iterator

```
MoveFirst([out, retval] VARIANT_BOOL*);
```
Moves to the first object or association in the iterator's collection. Returns false if empty

```
MoveNext([out, retval] VARIANT_BOOL*);
```
Moves to the next object or association in the iterator's collection. Returns false if last

```
Count([out, retval] long*);
```
Returns the number of objects or associations in the iterator's collection

```
Reset();
```
Reset the iterator to an uninitialised state (use New() or Find() functions to reinitialise)

```
Clone([out, retval] IDispatch**);
```
Creates a clone of this iterator

interface IRoleIterator

This interface is common to every composite role.

```
ObjectID([out, retval] BSTR*);
```
Returns the objectID GUID of the currently active object in the iterator

```
RoleType([out, retval] BSTR*);
```
Returns the name of the composite role of the currently active object in the iterator

interface I<CRname>

This is the interface of each composite role in the SIM.

```
//------------ Cast functions
CastAsCommon([out, retval] ICommonIterator**);
```
Returns same object as ICommonIterator

```
CastAsRole([out, retval] IRoleIterator**);
```
Returns same object as IRoleIterator

```
CastTo<SuperCRname>([out, retval] I<CRname>**);
```
Returns same object as I<SuperCRname>

```
CastTo<SubCRname>([out, retval] I<CRname>**);
```
Returns same object as I<SubCRname>

```
//------------ Creation functions
New([in] BSTR objectID);  // NOT if abstract
```
Prepares for the creation of a new role for an object with the specified objectID. If no objectID is specified, i.e., an empty string, then a new objectID will be generated.

```
Create();                 // NOT if abstract
```

Creates a new role that has been prepared with New()

```
Update();
```

Updates the currently active object

```
Drop();                    // NOT if abstract
```

Removes the current object from the composite role of the iterator

```
//------------ Search functions  // NOT if abstract
Find([in] BSTR condition,
     [out, retval] I<CRname>**);
```

The iterator will represent the collection defined by the argument string (ADO Filter)

```
FindID1([in] IRoleIterator* inObject,
              [out, retval] I<CRname>**);
```

Find the object with the same objectID as the argument object

```
FindID2([in] BSTR inObjectID,
        [out, retval] I<CRname>**);
```

Same as FindID1 except GUID argument

```
Find<PropertyName>([in] <ERtype> inValue,
                   [out, retval] I<CRname>**);
```

Find objects with the specified value for the specified property (attribute or collection).

```
Find<PropertyName>([in] IRoleIterator* inObject,
                   [out, retval] I<CRname>**);
```

Find objects associated to the specified object via the specified property (attribute or collection).

```
//------------ Attributes functions
Get<AttributeName>([out, retval] <AttributeType>*);
```

Returns the value of the specified attribute for the current object.

```
Set<AttributeName>([in] <AttributeType> inValue);
```

Sets the value of the specified attribute for the current object.

```
Get<AttributeName>([out, retval] I<CRname>**);
```

Returns the object referenced by the specified attribute of the current object.

```
Set<AttributeName>([in] IRoleIterator* inObject);
```

Sets the object referenced by the specified attribute of the current object.

```
Load<AttributeName>([in] BSTR inFileName);
```

Only for image typed attributes. Loads the content of a specified file into the specified image attribute of the current object.

```
Save<AttributeName>([in] BSTR inFileName);
```

Only for image typed attributes. Saves the content of the specified image attribute of the current object to file.

```
//------------ Collections functions
 (if collection is object)
Insert<CollectionName>([in] IRoleIterator* inObject);
```

Inserts the specified object into the specified collection of the current object.

```
Get<CollectionName>([out, retval] I<CRname>**);
```

Returns an iterator for the specified collection of objects.

```
Remove<CollectionName>([in] IRoleIterator* inObject);
```
Removes the specified object, or all if no argument object is specified, from the specified collection of the current object.

```
Count<CollectionName>([out, retval] long*);
```
Returns the number of objects in the specified collection of the current object.

```
//------------ Associations functions
GetAssos<RelationName>([out, retval] I<nDirAssName>**);
```
Returns an iterator for the collection of non-binary dir-associations (i.e., the specified relation) of the current object.

interface I<nBiAssName>

This is the interface of each bi-association in the SIM.

```
//------------ Cast functions
CastAsCommon([out, retval] ICommonIterator**);
```
Returns same object as ICommonIterator

```
//------------ Creation functions
New();
```
Prepares for the creation of a new association

```
Create();
```
Creates a new association that has been prepared with New()

```
Update();
```
Updates the currently active association

```
Drop();
```
Deletes the currently active association

```
//------------ Search functions
Find([in] BSTR condition,
            [out, retval] I<nBiAssName>**);
```
The iterator will represent the collection defined by the argument string (ADO Filter)

```
Find<AttributeName>([in] <ERtype> inValue,
                  [out, retval] I<nBiAssName>**);
```
Find associations with the specified value for the specified attribute.

```
Find<AttributeName>([in] IRoleIterator* inObject,
                  [out, retval] I<nBiAssName>**);
```
Find associations associated to the specified object via the specified attribute.

```
//------------ Attribute functions
Get<AttributeName>([out, retval] <AttributeType>*);
```
Returns the value of the specified attribute for the current association.

```
Set<AttributeName>([in] <ERtype> inValue);
```
Sets the value of the specified attribute for the current association.

```
Get<AttributeName>([out, retval] I<CRname>**);
```
Returns the object referenced by the specified attribute of the current association.

```
Set<AttributeName>([in] IRoleIterator* inObject);
```

Set the object referenced by the specified attribute of the current association.

```
Load<AttributeName>([in] BSTR inFileName);
```
Only for image typed attributes. Loads the content of a specified file into the specified image attribute of the current object.

```
Save<AttributeName>([in] BSTR inFileName);
```
Only for image typed attributes. Saves the content of the specified image attribute of the current object to file.

interface I<nDirAssName>

This is the interface of each dir-association in the SIM.

```
//------------ Cast functions
CastAsCommon([out, retval] ICommonIterator**);
```
Returns same object as ICommonIterator

```
//------------ Creation functions
New();
```
Prepares for the creation of a new association

```
Create();
```
Creates a new association that has been prepared with New()

```
Update();
```
Updates the currently active association

```
Drop();
```
Deletes the currently active association

```
//------------ Search functions
Find([in] BSTR condition,
          [out, retval] I<nDirAssName>**);
```
The iterator will represent the collection defined by the argument string (ADO Filter)

```
Find<AttributeName>([in] <ERtype> inValue,
                    [out, retval] I<nDirAssName>**);
```
Find associations with the specified value for the specified attribute.

```
Find<AttributeName>([in] IRoleIterator* inObject,
                    [out, retval] I<nDirAssName>**);
```
Find associations associated to the specified object via the specified attribute.

```
//------------ Attribute functions
Get<AttributeName>([out, retval] <AttributeType>*);
```
Returns the value of the specified attribute for the current association.

```
Set<AttributeName>([in] <ERtype> inValue);
```
Sets the value of the specified attribute for the current association.

```
Get<AttributeName>([out, retval] I<CRname>**);
```
Returns the object referenced by the specified attribute of the current association.

```
Set<AttributeName>([in] IRoleIterator* inObject);
```
Sets the object referenced by the specified attribute of the current association.

Load<AttributeName>([in] BSTR inFileName);
Only for image typed attributes. Loads the content of a specified file into the specified image attribute of the current object.

Save<AttributeName>([in] BSTR inFileName);
Only for image typed attributes. Saves the content of the specified image attribute of the current object to file.

2.3 SINAI Manager - The Prototype Tool GUI

SIVBSINAIManager is an ActiveX control, implemented in Visual Basic, that provides the graphical user interface of the SINAI prototype when run in Internet Explorer. The prototype front-end has not been prioritied - it is very crude and simple. You may prefer to include the various application and SIM cache components directly in your own software to use how it suits your purpose best.

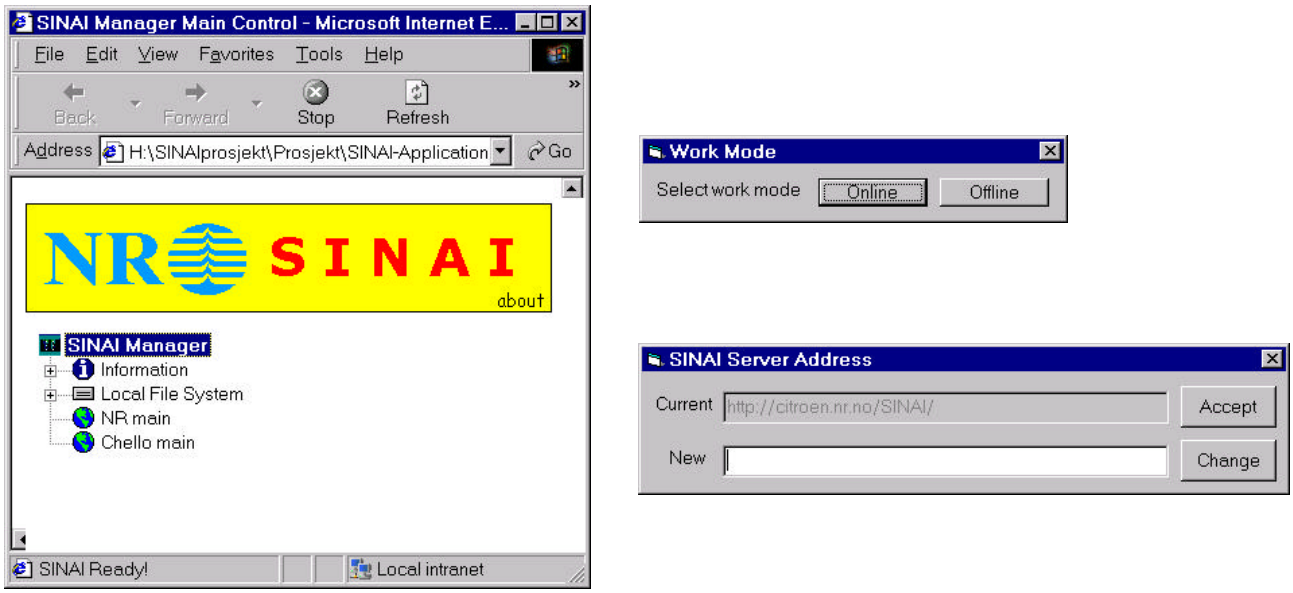

Figure 7. *Select work mode as offline or online, and if online then select default SINAI server*

SINAI Sources

After having started the application you can select whether to work online or offline. Offline means that all work will be performed locally without connecting to a SINAI server. Else, if you want to work online, then you must first select the initial SINAI server to contact. The client then contacts this server and retrieves information on all known SINAI sources. A SINAI source is a SINAI compliant server that offers the basic SINAI services of this prototype. Figure 8 below illustrates the (very simple!) SIM, called SIMSource, for SINAI source information.

In figure 7 the client got information about two sources, each illustrated by a "world" icon, namely "NR main" and "Chello main".

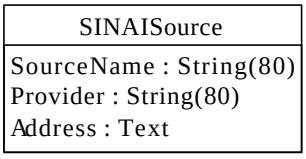

Figure 8. *The SIMSource information model for information on SINAI sources; i.e., SINAI compatible web sites.*

Browse Files

By selecting one of the information sources under the "Information" icon, or double clicking this icon, a browser is started so that you can search for documents either locally or on the web.

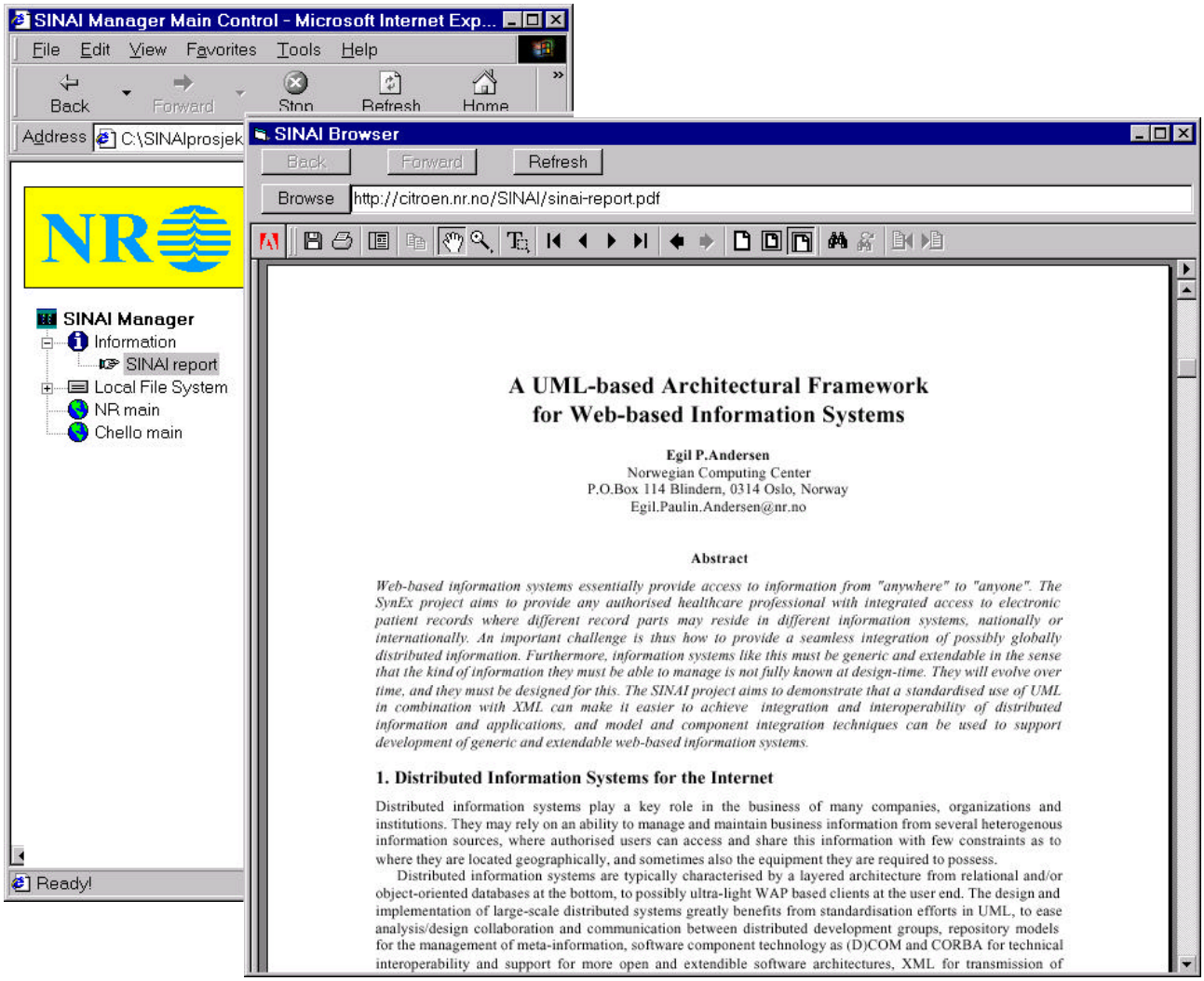

Figure 9. *Browser for web information*

Log on to SINAI Source

You can start working with information on one of the SINAI sources by selecting "Log on" from the menu. In the current test version any user name and password will be accepted.

After a successful log on, and similarly in the Local File System for offline work, you will get three nodes in the three-view control. The "SINAI Information Models" will represent a SIM cache with meta-information about SIMs; the "SINAI Application Models" similarly represents a SIM cache with meta-information about SAMs; and "RDB Schema" represents a SIM cache with meta-information about relational database schemas.

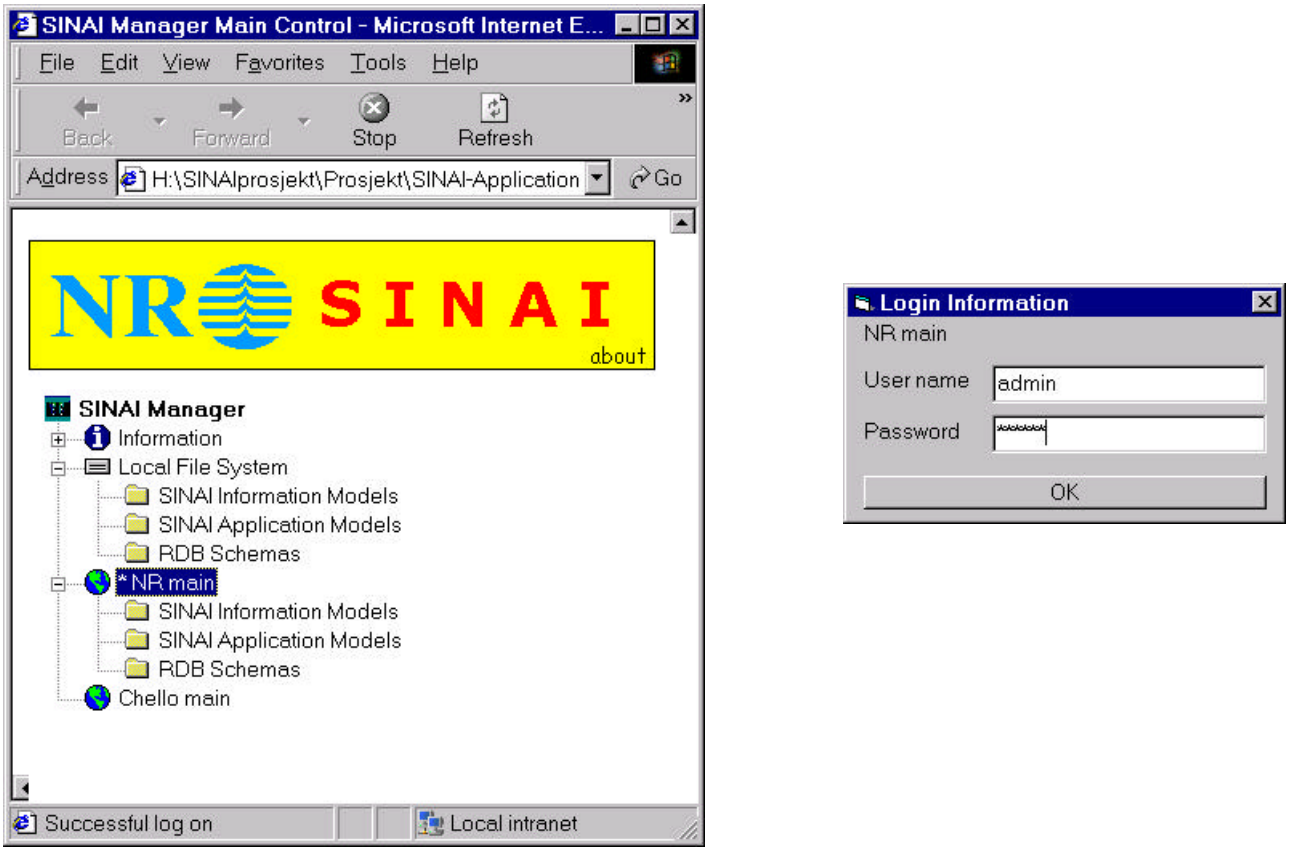

Figure 10. *Logging on to a SINAI source*

Load Caches

The first operation to do when working with the prototype is to load the caches that you need. There are three alternatives.

- Load a cache from a specified file
- Load an empty cache from the specified source
- Load all information from the specified source

Figure 11, to the left, illustrates the result after loading all SIM information from the "NR main" source. Each SIM in the successfully loaded cache is represented by a separate icon under "SINAI Information Models". You may then continue and work with these models; e.g. generate a Rose model for them, or generate an RDB schema, as illustrated to the right in figure 11.

The rest of this section describes the various operations that can be performed with the current prototype.

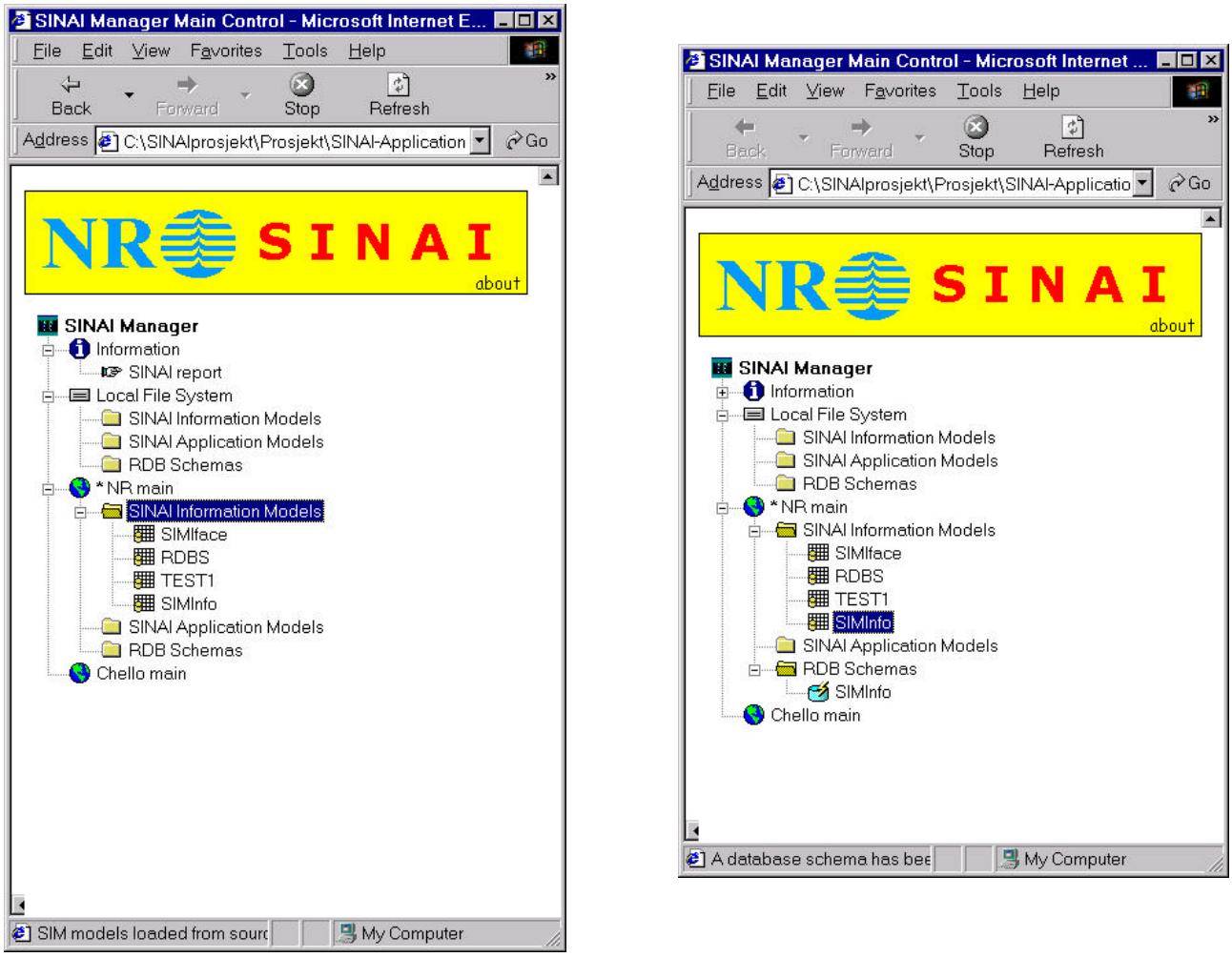

Figure 11. *Load information about SIMs from the "NR main" source*

Cache Menu Operations

The following operations are available on three different types of SIM caches in the prototype:

- SINAI Information Models:
- Create a new SIM based on a Rational Rose model
- Verify that a Rational Rose model satisfies the SIM modeling conventions
- Load an empty cache from this source
- Load all the SIM models from this source
- Save the current cache to this source
- Load a cache from file
- Save the current cache to file

- SINAI Application Models:
- Load an empty cache from this source
- Load all the SAM models from this source
- Save the current cache to this source
- Load a cache from file
- Save the current cache to file

- Relational Database Schemas:
- Load an empty cache from this source

- Load all the RDB Schemas from this source
- Save the current cache to this source
- Load a cache from file
- Save the current cache to file

SAM and SAM Model Menu Operations

The following operations are available for each SIM model:
- View a picture of this SIM (a Rational Rose image)
- Generate a Rose model for this SIM
- Generate a cache for this SIM (IDL+Visual Basic implementation)
- Generate an RDB Schema for this SIM.
  The generated schema is inserted into the RDBSchema cache of the same source.
- Delete the selected SIM

No operations are currently available for SAMs, but the following operations will be added once the SAM tools are ready:
- View a picture of this SAM (a Rational Rose image)
- Generate a Rose model for this SAM
- Generate IDL interfaces for this SAM
- Delete the selected SAM

RDB Schema Menu Operations

The following operations are available for each relational database schema:
- Generate a Microsoft SQL Server script to install this RDB Schema.
- Delete the selected RDB Schema

3. Project Files and Organisation

3.1 SINAI Development

The following is an outline of the directories and files that together constitute the implementation of the SINAI prototype.

- SINAI-RoseModel - SINAI Rose models
- SINAI-Database - several database schemas for the prototype
- SINAI-Application - a set of directories each containing a SINAI application component
- SINAI-Interfaces - IDL interfaces and compiled type libraries (ready to be registered) for prototype components
- SINAI-Utility - various basic SINAI application components used in most SINAI projects
- SINAI-SystemFiles - XML template files for the code generation
- SINAI-ErrorHandling - an overview of error codes and messages for the prototype
- SINAIwebserver - files to be placed on webservers offering server access to the prototype
- SINAI-MIDL - Visual C++ project used as an IDL compiler (for simplicity)

The following sections describe the content of these directories in more detail.

SINAI-RoseModel

There are 5 SIM Rose models in the current SINAI prototype, namely

- sinai-siminfo-0.1.mdl - The SIM meta-model in figure 2 above. This model will be referred to as the *"SIMInfo"* model.
- sinai-simiface-0.1.mdl - An alternative SIM meta-model used for temporary storage of SIM meta-information during code generation; shown in figure 14 below. This model will be referred to as the *"SIMIface"* model.
- sinai-simsource-0.1.mdl - A SIM used for storing information on SINAI *sources* (i.e., SINAI compliant web sites); shown in figure 8 above. This model will be referred to as the *"SIMSource"* model.
- sinai-rdbs-0.1.mdl - The RDBS meta-model in figure 4 above. This model will be referred to as the *"RDBS"* model.
- sinai-sam-0.2.mdl - The SAM meta-model in figure 3 above. This model will be referred to as the *"SAM"* model.

SINAI-Database

There are 4 SQL Server database schemas in the prototype, corresponding to the above SIM models except SIMIface which is only used for temporary information. They exist within the following directories:

- SINAI-SIMInfo-DB - SQL Server schema for the SIMInfo model.
- SINAI-SIMSource-DB - SQL Server schema for the SIMSource model.
- SINAI-RDBS-DB - SQL Server schema for the RDBS model.
- SINAI-SAM-DB - SQL Server schema for the SAM model.

In addition the directory *"SINAI-DB-DataType"* contains the SQL Server data type declaration file *"sinai-db-datatype.sql"*, and these declarations must be installed in every SINAI SQL Server database.

SINAI-Application

- SIVBSIMInfoCache - Cache components for the SIMInfo model; cf figure 2
- SIVBSIMIfaceCache - Cache components for the SIMIface model; cf figure 14
- SIVBSIMSourceCache - Cache components for the SIMSource model; cf figure 8
- SIVBRDBSCache - Cache components for the RDBS model; cf figure 4
- SIVBSAMCache - Cache components for the SAM model; cf figure 3
- SIVBSIMInfo2Iface - Component for converting a SIMInfo cache to SIMIface cache
- SIVBGenericSource - Component with services for using Server stored procedures via OLE DB providers
- SIVBCacheXMLFile - XML and file system services used by SIM cache components
- SIVBGenerateRoseModel - Component with services for generating Rose models from SIM model information in a SIM meta-model database.
- SIVBStoreRoseModel - Component with services for validating and storing Rose models in a SIM meta-model database.
- SIVBGenerateCache - Component with services for generating IDL specifications and/or ADO-based Visual Basic components for a SIM cache.
- SIVBGenerateRDB - Component with services for generating a relational database schema in the RDBS model based on a SIMInfo model, or generating an SQL Server script for creating a database schema as defined by an RDBS model.
- SIVBSINAIServer - Component with the web-services used by the SINAI prototype. The SOAP Toolkit v.2.0 is used to make the services of this component available as SOAP services on a webserver (using ISAPI).
- SIVBSaveCacheServer - This component is made to handle what seems like a bug in the SOAP Toolkit when transferring large data sets (a cache) from the client to the server.
- SIASPSINAIWebServer - An ASP script as the server front-end for SIVBSaveCacheServer
- SIVBSINAIServerProxy - A client front-end to the web-services offered by SIVBSINAIServer (and SIVBSaveCacheManager due to a SOAP Toolkit problem).
- SIVBSINAIManager - ActiveX component that is the GUI of the SINAI prototype

SINAI-Interfaces

The following are IDL defined interfaces for some of the above application components:

- SIIfaceCacheManager - model-independent IDL interfaces for cache components
- SIIfaceSIMInfoCache - IDL interfaces for SIMInfo cache components
- SIIfaceSIMIfaceCache - IDL interfaces for SIMIface cache components.
- SIIfaceSIMSourceCache - IDL interfaces for SIMSource cache components
- SIIfaceRDBSCache - IDL interfaces for RDBS cache components
- SIIfaceSAMCache - IDL interfaces for SAM cache components
- SIIfaceGenericSource - IDL interfaces for the SIVBGenericSource component
- SIIfaceBasicTypeCache - made for a future version of the prototype

SINAI-Utility

The following are some components that will be common to many SINAI projects:

- SIVCClientGUIDmanager - Component for generating new GUID's (Visual C++ - currently used)

- SIVBClientGUIDManager - Component for generating new GUID's (Visual Basic - not used)
- SIVBServerGUIDManager - Component for generating new GUID's (Visual Basic - not used)
- SIVBClientTrace - Component that writes traces of errors and warnings
- SIVBServerTrace - Same as previous, but runs under MTS
- SIVBFileManager - Component offering some common file system services
- SIVBServerFileManager - Same as previous, but runs under MTS

SINAI-SystemFiles

Here there is one directory *"CodeTemplates"* with the following sub-directories for XML template files for code generation:

- IDLTemplates - "idl-templates.xml" for SIM cache IDL interface generation
- VBADOTemplates - "vbado-templates.xml" for Visual Basic SIM cache component generation
- RDBTemplates - "rdb-templates.xml" for SQL Server schema generation

An overview of the SINAI Prototype Components

Figure 12 illustrates an overview of the main constituents and components of the SINAI prototype, and roughly how they relate to each other.

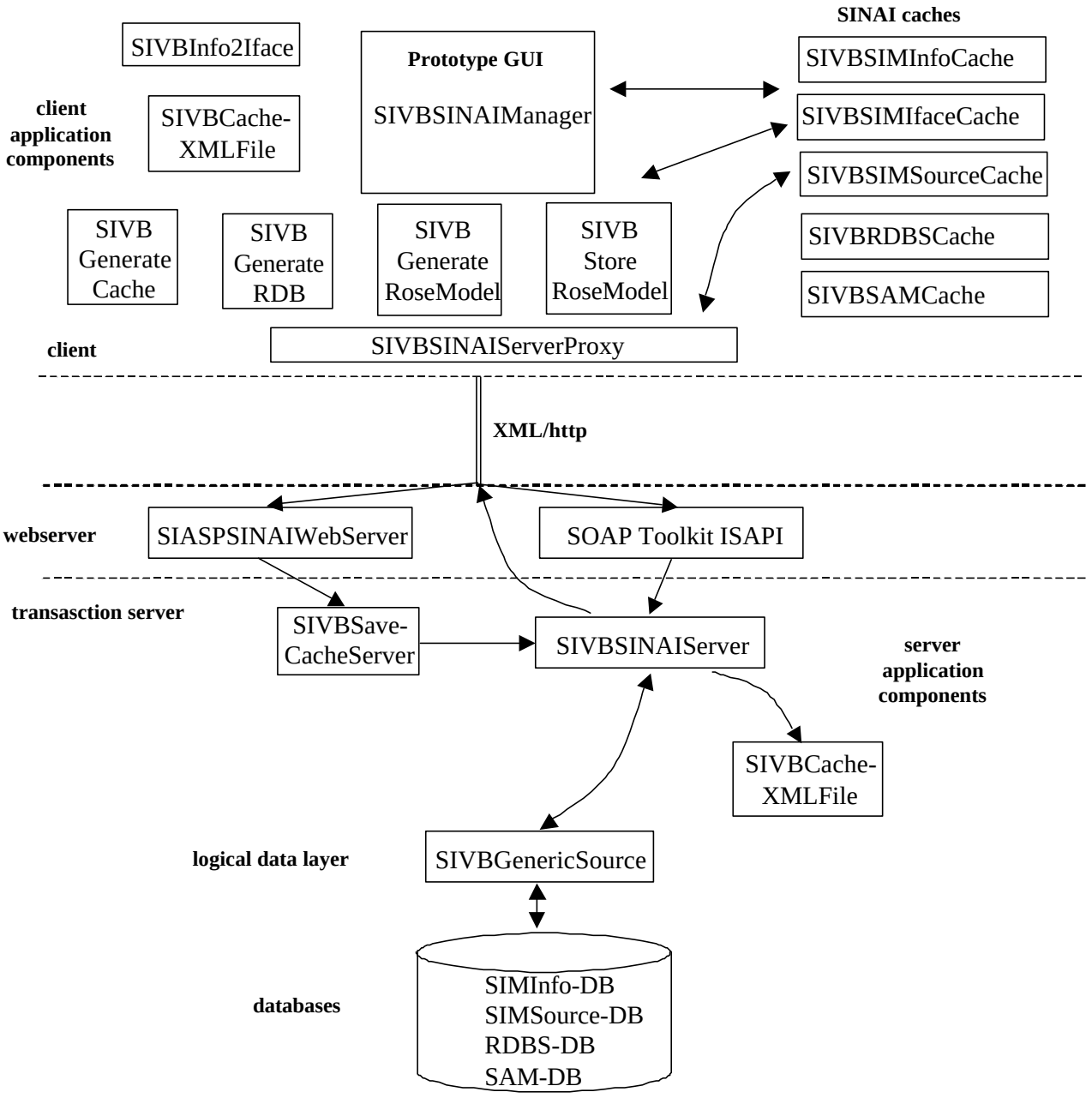

Figure 12. *An overview of components in the SINAI prototype.*

3.2 SINAI Runtime

The prototype expects the following directories at runtime

C:\SINAIruntime
C:\SINAIruntime\LogFiles

and the following files, which are empty caches for the SINAI information models SIMInfo, SIMSource and RDBS:

C:\SINAIruntime\siminfo-empty-cache.xml
C:\SINAIruntime\simsource-empty-cache.xml
C:\SINAIruntime\rdbs-empty-cache.xml

It may also be useful to include the XML template files in the "SINAI-SystemFiles\CodeTemplates\" directory above since these are used for the different kinds of SINAI code generation and thus must be selected for these operations.

The following file (if it exists) contains information on which SINAI server is considered this clients default server (a user must explicitly accept this initially when executing the prototype):

C:\SINAIruntime\defaultserver.txt

The "C:\SINAIruntime\LogFiles" directory may contain the following files in case any errors has occurred on either the client or the server, respectively:

C:\SINAIruntime\LogFiles\clienttrace.log
C:\SINAIruntime\LogFiles\servertrace.log

4. Store and Generate Rational Rose Models for SIM

The SINAI components that store and generate Rose models to/from a SIM meta-model database, i.e., *SIVBStoreRoseModel* and *SIVBGenerateRoseModel*, respectively, both make straightforward use of Rose' COM interfaces.

However, notice that it is very important to properly manage the object identities (GUIDs) of Rose models and model entities, versus the identities of IDL interfaces, Visual Basic components, database object entries, etc. Hence, unless the Rose modeller does this himself, when storing a Rose model then the "Store Rose Model" service will assign a unique identity (a GUID) to the Rose model itself, and every composite role, association and attribute within this model. Figure 13 illustrates the format of such a specification, and notice that the "Documentation" property of Rose entities are used for this.

The "Generate Rose Model" service will correspondingly insert GUID identifiers like this based on the identities recorded in the SIM meta-model database.

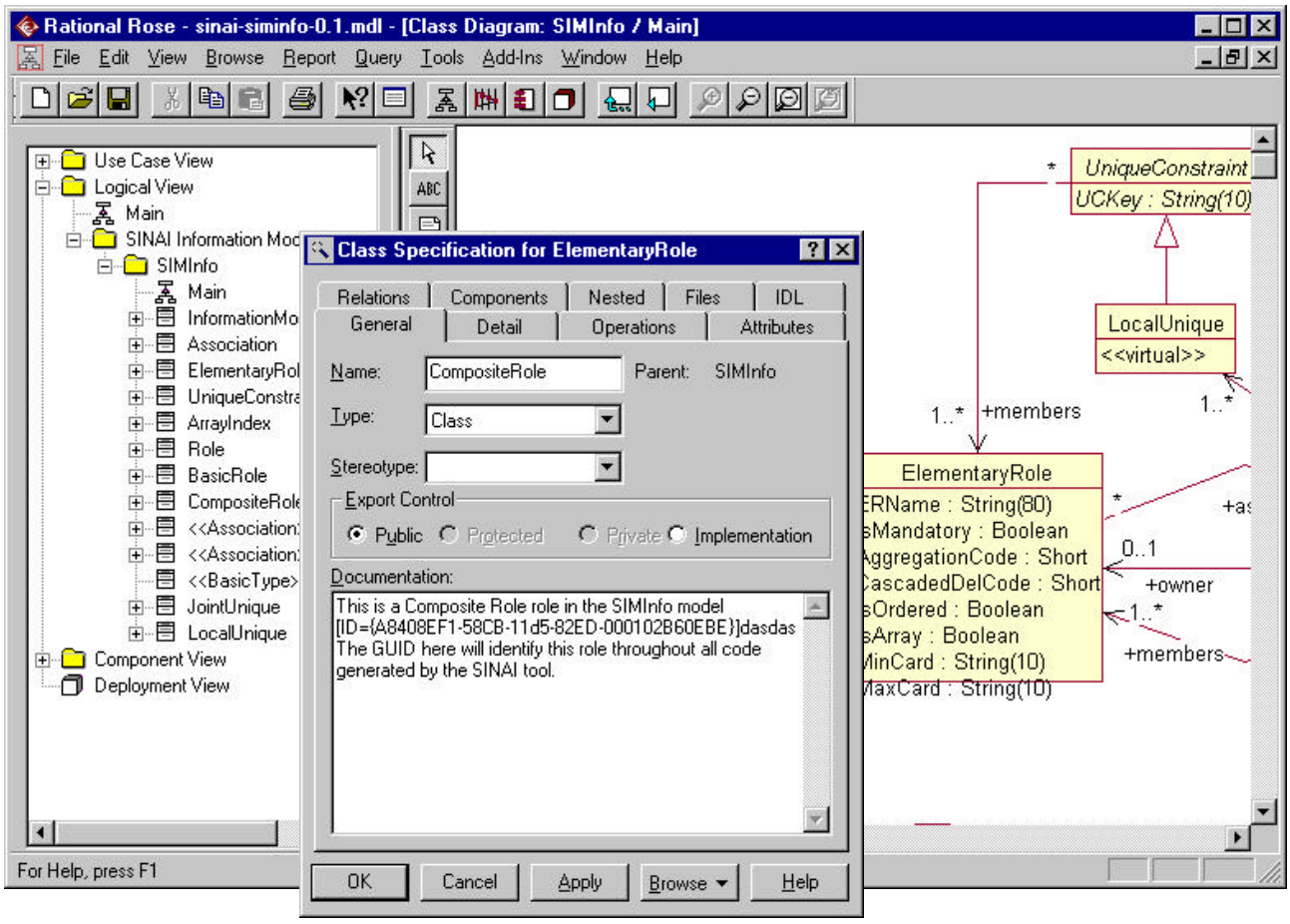

Figure 13. *Every Rose model, composite role, association and attribute is assigned a unique GUID when being stored in a SIM meta-model database.*

SIVBStoreRoseModel.IStoreRoseModel interface

```
Public Function VerifyRoseModel(ByVal inModelFile As String, _
                                ByVal inErrorCatalog As String, _
                                ByVal inErrorFile As String) As Boolean
```

Rose model verification with respect to a number of requirements to legal SIM models.

inModelFile is the Rose model file, inErrorCatalog and inErrorFile the directory and file to put the report.

```
Public Sub StoreRoseModel(ByVal inSIMCache As SIIfaceSIMInfoCacheLib.-
                                              ISINAIInformationModel, _
                          ByVal inModelFile As String)
```

Store a Rose model in the file inModelFile in the SIMInfo cache referenced by the inSIMCache reference. Notice that the SIMInfo cache referenced by inSIMCache must later on be saved to a SIM meta-model database to persist this model information.

SIVBGenerateRoseModel.IGenerateRoseModel interface

```
Public Sub GenerateRoseSIM(ByVal inSIMCache As SIIfaceSIMInfoCacheLib.-
                                               ISINAIInformationModel, _
                           ByVal inModelID As String, _
                           ByVal inModelFile As String, _
                           ByVal inOverwriteFile As Boolean)
```

Based on the information of a particular SIM with the modelID inModelID in a SIMInfo cache referenced by inSIMCache, create a SIM Rose model in the file inModelFile. If inOverwriteFile is false then no model will be created if the file inModelFile already exists.

5. Store and Generate Rational Rose Models for SAM

Services for verifying, storing or generating Rose models based on SINAI Application Models (SAMs) has not yet been implemented for the SINAI prototype.

These services and the functions for invoking them will be very similar to the above services for verifying, storing and generating such models based on SIMs. This of course except for the kind of Rose concepts used and generated.

6. Generate IDL Specifications for SIM

SIVBGenerateCache

The SIVBGenerateCache component, which offers a service for generating SIM cache IDL interface specifications is the same component which also offers a service for generating Visual Basic component code for implementing these interfaces.

SIVBGenerateCache implements a single interface, and the following functions are used to generate SIM cache IDL:

```
Public Sub ComputeModelInformation(ByVal inSIMInfoCache As SIIfaceSIMInfoCacheLib.-
                                                    ISINAIInformationModel, _
                                   ByVal inCacheName As String, _
                                   ByVal inModelID As String)
```

This function is used to convert information in a SIMInfo cache, referenced by inSIMInfoCache, into a SIMIface cache (see below).
The inCacheName argument specifies the short name to use for the cache (e.g. like SIMInfo, SIMIface, etc), while inModelID identifies the SIM in inSIMInfoCache for which to generate the IDL.

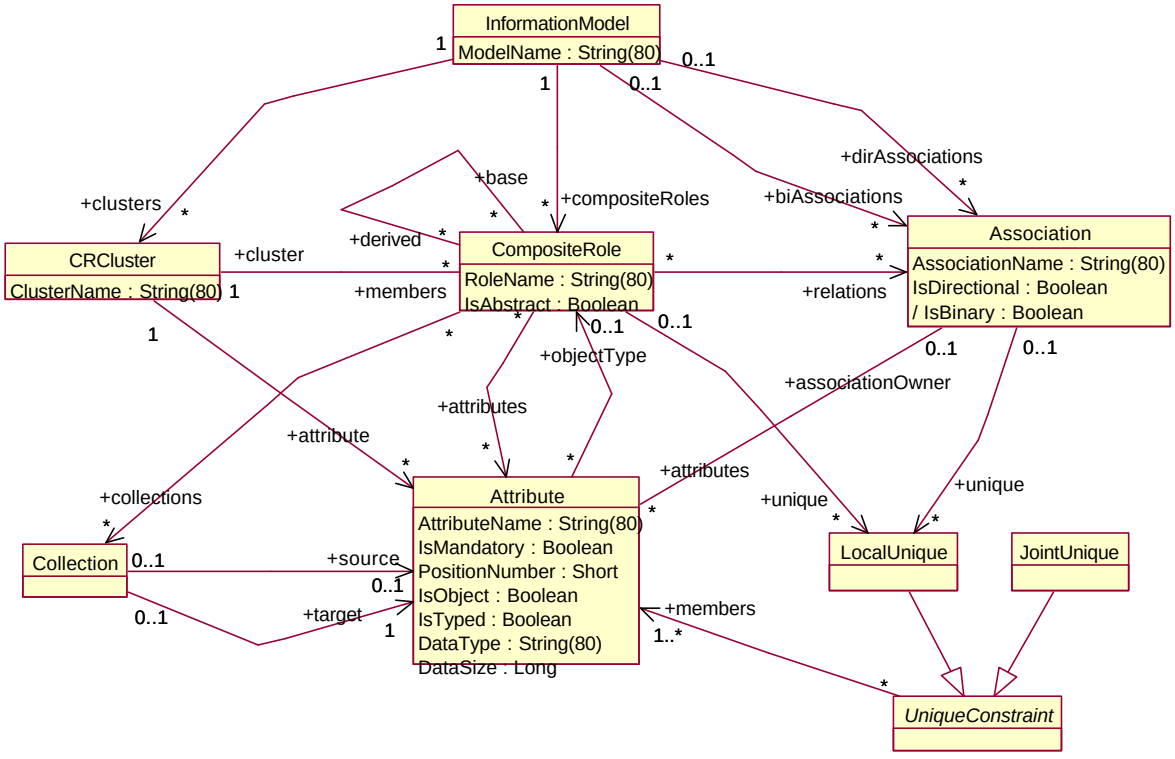

Figure 14. *An alternative meta-model, called SIMIface, for SINAI Information Models which is useful as an alternative to the SIMInfo model in figure 2 when generating IDL specifications and SIM cache components.*

```
Public Sub GenerateCacheIDL(ByVal inTemplateCatalog As String, _
                            ByVal inIDLCatalog As String, _
                            inoutInterfaceGUID As String)
```

This function performs the actual generation by using the IDL XML templates (see below). inTemplateCatalog identifies the catalog where to find the XML template, and inIDLCatalog identifies the catalog to put the generated IDL. inoutInterfaceGUID can be used to specify the

GUID of the type library generated in the IDL. Alternatively, if inoutInterfaceGUID is an empty string then a new GUID will be used and returned to the caller via this argument.

The SIM cache interfaces that are generated are described in section 2.2.3 above. They are generated based on SIM information in the SIMInfo cache (figure 2), but the SIM in figure 14, called SIMIface, is used for the actual generation. That is, information in a SIMInfo cache is transformed into a SIMIface cache, and this is the task of the ComputeModelInformation() function above.

XML Templates for generating IDL

All code generation in the SINAI prototype is done by using XML templates. That is, different parts of the code files to be generated are coded in XML, i.e., as text within XML elements. Parts that depend on e.g. the SIM model used as source for the generation will be replaced with other XML elements that will be replaced (substituted) during code generation with proper names or text.

Section B.1 in the appendix presents the XML template for SIM cache IDL generation. For example, the following XML

```
<GetSetValue> <!-- ==== START ========================================== -->
         [id(<UFID/>), helpstring("Function Get<PropertyName/>")]
         HRESULT Get<PropertyName/>([out, retval] <DataType/>*);

         [id(<UFID/>), helpstring("Function Set<PropertyName/>")]
         HRESULT Set<PropertyName/>([in] <DataType/> inValue);
</GetSetValue> <!-- ==== END ============================================ -->
```

is used to generate IDL for get/set functions of composite role and association attributes when they are basic value types (i.e., not object references). There are three XML tags to substitute, namely <UFID/>, which is a number unique within the interface to which these functions belong; <PropertyName/> which is the name of the attribute; and finally <DataType/> which is the IDL data type for this attribute.

The XML created by making these substitutions is then inserted into the XML document for the overall IDL interface in a particular position identified by an XML element <AttributeFunctions/>.

The usefulness of automatically generated code should not be overestimated; e.g., complex business rules usually require non-trivial manual intervention. However, the code generated in SINAI, both the model-specific as well as the predefined code, is such that manual customisation, optimisation and specialisation is possible everywhere. The fact that all the SINAI code templates stems from similar XML template files makes it very easy to do manual additions to the generated code without involving the programs that does the actual generation. Thus this use of XML makes for a very flexible code-generation approach. It may be worthwhile to study this further in a separate project.

Notice that the current prototype does *not* include support for preserving manual additions and changes when regenerating code, but such mechanisms can be included in our XML-based code generation approach without undue effort.

7. Generate IDL Specifications for SAM

Services for generating IDL specifications for SINAI Application Models (SAMs) has not yet been implemented for the SINAI prototype.

These services and the functions for invoking them will be very similar to the above services for generating IDL specifications based on SIMs; i.e., similar XML IDL specification templates will be used.

8. Generate Visual Basic Cache Components for SIM

SIVBGenerateCache

The SIVBGenerateCache component, which offers a service for generating Visual Basic component code for implementing a SIM cache is the same component which also offers a service for generating the IDL interface specification of this cache.

SIVBGenerateCache implements a single interface, and the following functions are used to generate a Visual Basic component to implement the corresponding IDL of a SIM cache:

```
Public Sub ComputeModelInformation(ByVal inSIMInfoCache As SIIfaceSIMInfoCacheLib.-
                                                    ISINAIInformationModel, _
                                   ByVal inCacheName As String, _
                                   ByVal inModelID As String)
```

This function is used to convert information in a SIMInfo cache, referenced by inSIMInfoCache, into a SIMIface cache (see IDL generation above).
The inCacheName argument specifies the short name to use for the cache (e.g. like SIMInfo, SIMIface, etc), while inModelID identifies the SIM in inSIMInfoCache for which to generate the IDL.

```
Public Sub GenerateCacheVBADO(ByVal inTemplateCatalog As String, _
                              ByVal inCodeCatalog As String, _
                              inInterfaceGUID As String)
```

This function performs the actual generation by using the VBADO XML templates (see section 8.2 in the appendix).
inTemplateCatalog identifies the catalog where to find the XML template, and inCodeCatalog identifies the catalog to put the generated Visual basic project and its files.
Notice: The inInterfaceGUID argument should be the same GUID as the "inoutInterfaceGUID" argument when generating IDL. This is the GUID for the type library generated for this IDL, and the generated VB will make a reference to it if provided here. Otherwise the IDL type library reference must be added by hand before compiling the generated VB project.

Usually both the SIM cache IDL and the implementation code will be generated at the same time, and in that case it suffice with a single call to ComputeModelInformation() before calling GenerateCacheIDL() and then GenerateCacheVBADO (or the other way around for the latter two).

Cache Isolation

The current cache implementation does not support multiple simultaneous transactions, but it supports committ/rollback transactional properties. This can be very useful to avoid inconsistent cache state if large updates (e.g. a client refresh received from the server) suddenly fail.

The _SCode attribute of SIM cache recordsets is used to handle isolation for operations performed by any ongoing transaction, and the following are states defined by different values for the _SCode attribute:

_SCode = 0	Unchanged information
_SCode = 1, 3	Updated information
_SCode = 2	Committed new information
_SCode = 5, 6, 7, 8	Uncommitted deleted information
_SCode = 12	Uncommitted new information
_SCode = 15, 16, 17, 18	Committed deleted information

Notice that _SCode < 10 will make information "visible" in the cache, whilc _SCode >= 10 makes it "non-existing".
The rules for updating _SCode is as follows:

- An update of unchanged, committed new or uncommitted deleted information, increase _SCode with 1 (+1)
- A commit of uncommitted new information, decrease _SCode with 10 (-10)
- A commit of uncommitted deleted information, increase _SCode with 10 (+10)
- An uncommitted delete of unchanged, updated or committed new information, increase _SCode with 5 (+5)
- A rollback of uncommitted deleted information, decrease _SCode with 5 (-5)
- A rollback of uncommitted new information, remove the information from the cache

Data Type Conversion

The following table shows the data type mappings used in the SINAI prototype.

SINAI Models	Visual Basic	Visual C++	IDL	SQL Server	OLE DB/ADO
Boolean	Boolean	boolean	VARIANT_BOOL	Bit	adBoolean
Short	Integer	short	short	SmallInt	adSmallInt
Long	Long	long	long	Int	adInteger
Double	Double	double	double	Numeric	adNumeric
Float	Single	float	float	Numeric	adNumeric
String(n) (<256)	String	CComBSTR	BSTR	VarChar(n)	adVarChar(n)
Text	String	CComBSTR	BSTR	Text	adVarChar(n)
Image	Variant	CComVariant	VARIANT	Image	adVarBinary(n)
DateTime	Date	???	DATE	DateTime	adDBTimeStamp
GUID	Variant/String	CComBSTR	BSTR	UniqueIdentifier	adGUID

Notice: If no explicit size is defined for OLE DB/ADO for types Text and Image, then a maximum size of 2147483647 (2^31) can be used.

9. Generate Relational Database Schemas from SIM

SIVBGenerateRDB

The SIVBGenerateRDB component implements a single interface, and the following functions are used to generate a relational database schema based on a particular SIM:

```
Public Sub ComputeRDBSchema(ByVal inSIMInfoCache As SIIfaceSIMInfoCacheLib.-
                                                    ISINAIInformationModel, _
                            ByVal inModelID As String, _
                            outRDBSCache As SIIfaceRDBSCacheLib.ISINAIInformationModel
                            outDBSchemaID As String)
```

This function is used to convert information about a particular SIM, identified by inModelID, in a SIMInfo cache (see figure 2), referenced by inSIMInfoCache, into a RDBS cache (see figure 4).
outDBSchemaID will return the object ID of the RDB schema generated in the RDBS cache.

```
Public Sub GenerateSSSchemaFile(ByVal inRDBSCache As SIIfaceRDBSCacheLib.-
                                                     ISINAIInformationModel, _
                                ByVal inDBSchemaID As String, _
                                ByVal inTemplateCatalog As String, _
                                ByVal inRDBCatalog As String, _
                                ByVal inRDBSchemaPrefix As String)
```

This function performs the actual generation by using the VBADO XML templates (see section 8.3 in the appendix).
The inRDBSCache is a reference to an RDBSchema cache with information on the schema to generate. inDBSchemaID is the ID of the schema in inRDBSCache to generate.
inTemplateCatalog identifies the catalog where to find the XML template, and inRDBCatalog identifies the catalog to put the generated SQL script.
inRDBSchemaPrefix is the prefix that will be used to identify all schema objects uniquely within a particular database. This is necessary since several e.g. composite roles in different SIMs may have the same name and also have their database tables installed in the same database.

Keys, Indexes and Uniqueness Constraints

The current prototype generate primary keys, foreign keys and unique indexes. Joint-unique constraints are not currently implemented, but there are no principal problems in this. They can be enforced either by indexes or via relatively simple triggers, depending on the case.

RDB Stored Procedures

When designing and implementing a relational database schema, then there are several good reasons for always encapsulating the database tables behind an "interface" of *stored procedures*.

- Constraints and business rules that are inherently linked to the database schema, independent of which application uses the database, should be enforced within the database, and stored procedures may be the only means for achieving this.
- The tables of a database schema are often subject to minor changes, e.g. for performance reasons, but such changes should be transparent to the application.
  For example, if a database implements a UML model in a "one-to-one" manner, then the stored procedure interface would contain procedures for creating, updating, deleting objects of each class, amongst others. However, due to the inherent "mismatch" between a relational schema with tables and an object-oriented UML model, it may well be necessary to implement different inheritance hierarchies in the UML model differently with respect to table layout, just depending on the individual characteristics of each inheritance hierarchy (see below).

- Improved performance since executing a set of SQL statements within a stored procedure is often more efficient than executing SQL statements, as requested by application components, individually.

Notice that the current version of the prototype does not generate stored procedures, but the XML template is ready for this, and the following is a list of stored procedures that will be generated for a particular SIM.

- For each composite role name <CR> in the SIM:

```
SP_<prefix>_Create<CR>(objectid:GUID, <param each attribute>)
SP_<prefix>_Drop<CR>(objectid:GUID)
SP_<prefix>_Update<CR>(objectid:GUID, <param each attribute>)
```

- For each binary bi-association named <Assos> in the SIM:

```
SP_<prefix>_Create<Assos>(<target1 value>, <target2 value>)
SP_<prefix>_Drop<Assos>(<target1 value>, <target2 value>)
            <target1 value> and/or <target2 value> may be Null, and in that case
            a delete will be made independent of the Null specified column
SP_<prefix>_Update<Assos>(<oldtarget1 value>, <newtarget1 value>,
                          <oldtarget2 value>, <newtarget2 value>)
```

- For each binary dir-association named <Assos> in the SIM:

```
SP_<Model>_Create<Assos>(ownerid:GUID, <target value>)
SP_<prefix>_Drop<Assos>(ownerid:GUID, <target value>)
             <target value> is not mandatory. If Null then every association with
           the specified owner will be deleted.
SP_<prefix>_Update<Assos>(ownerid:GUID, <oldtarget value>, <newtarget value>)
```

- For each non-binary bi-association named <Assos> in the SIM:

```
SP_<prefix>_Create<Assos>(assosid:GUID, <param each attribute>)
SP_<prefix>_Drop<Assos>(assosid:GUID)
SP_<prefix>_Update<Assos>(assosid:GUID, <param each attribute>)
```

- For each non-binary dir-association named <Assos> in the SIM:

```
SP_<prefix>_Create<Assos>(assosid:GUID, ownerid:GUID, <param each attribute>)
SP_<prefix>_Drop<Assos>(assosid:GUID)
SP_<prefix>_Update<Assos>(assosid:GUID, <param each attribute>)
```

Notice: The *SIVBGenericSource* component is useful to invoke arbitrary stored procedures in a database server (made available via some OLE DB provider). Each parameter of the stored procedure to call can be set individually, and output parameters can be collected afterwards. If the procedure return results via select statements then these results can be received as either ADO recordsets, or alternatively in SINAI XML format.

Implementing Inheritance Hierarchies in Relational Database Schemas

Figure 15 illustrates some of the most common alternatives for implementing object-oriented inheritance relationships in relational database schemas. Alternative d), i.e., one full table for each composite role in a SIM, has been used in the current version of the prototype. This would normally not be the preferred choice for traditional object-oriented class models, but, for reasons explained in the other SINAI report [1], there is a difference here since SIMs contain roles and not classes. However, the SINAI specification allows for a modeller to specify alternative e), with the "IsVirtual" modelling construct, which is often also the best relational alternative for traditional class-based models.

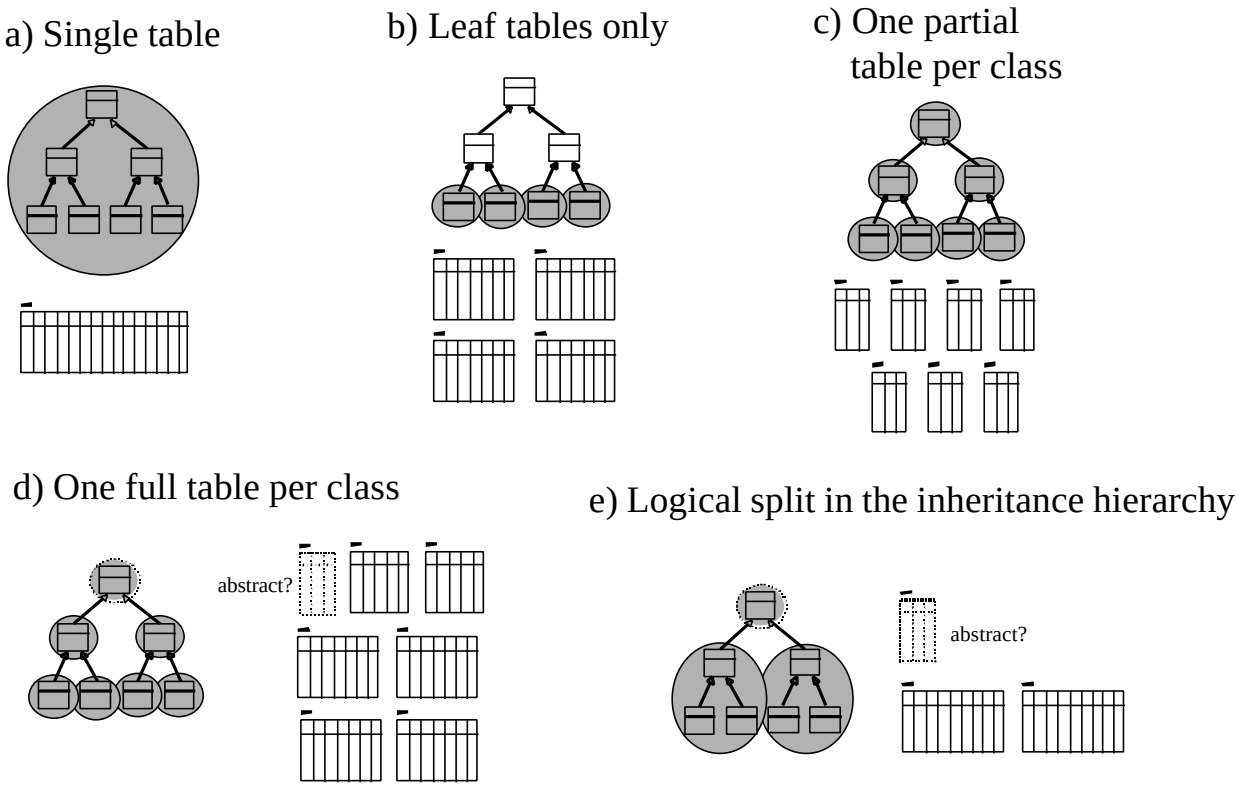

Figure 15. *Relational implementation alternatives for object-oriented inheritance hierarchies.*

10. Client-Server Interaction with SOAP and XML over http

SINAI XML

We have not spent much time so far to decide for an XML format for SINAI. This should preferably be some standardised XML format for UML; e.g. from XMI [25]. Instead we have used ADO XML for object information, and SOAP (via Microsoft's SOAP Toolkit) for object function invocation. This independent of which SIMs or SAMs that are involved.

Appendix A shows an example of ADO XML.

SIVBSINAIServer and SIVBSINAIServerProxy

SIVBSINAIServer implements a server-side component that offers the services used by the current SINAI prototype. These services are made available as web services, accessible via SOAP (and using the ISAPI option for the web server integration).

We have used the Microsoft SOAP Toolkit v.2.0 [17] for the prototype. Clients can access SOAP servers as if they access any other COM component by using the Toolkit's client component. However, we experienced problems with the Toolkit when trying to transfer relatively large volumes of data from the client to the server. For example, when trying to send XML for a SIM cache with model information about SIMInfo amongst others, then a fatal error occured for some as yet unknown reason. Therefore we have implemented an ad-hoc non-SOAP (but XML over http) solution to transfer large amounts of data from a SINAI client to the server. This solution uses the SIASPSINAIWebServer ASP script, and the SIVBSaveCacheServer component. However, to make this SOAP Toolkit fix transparent to SINAI clients we have implemented a client-side component SIVBSINAIServerProxy which offers the same services as the SIVBSINAIServer component.

The following functions, and web services, are offered by SIVBSINAIServer:

```
Public Function LogOn(ByVal inUserName As String, _
                      ByVal inPassword As String) As String
```

If successful then LogOn returns a GUID as a UserID for the client to use in further interaction.

NOTICE: The current SINAI prototype does not support authentication or access control!

```
Public Sub LogOff(ByVal inUserID As String)
```

```
Public Function LoadSINAISources(ByVal inUserID As String) As String
```

Returns SINAI XML for a SIMSource cache with every source known to the SINAI server.

```
Public Function LoadAllSIMs(ByVal inUserID As String) As String
```

Returns SINAI XML for a SIMInfo cache with all SIMs stored on the SINAI server.

```
Public Function LoadSIM(ByVal inUserID As String, _
                        ByVal inModelID As String) As String
```

Returns SINAI XML for a SIMInfo cache with information on the SIM identified by the inModelID argument.

```
Public Function SaveSIMCache(ByVal inUserID As String, _
                             ByVal inCacheXML As String) As Boolean
```

inCacheXML is a SINAI XML representation of a SIMInfo cache, and the content of this cache will be stored in the default SINAI SIMInfo database on this server.

```
Public Function LoadAllRDBSs(ByVal inUserID As String) As String
```

Returns SINAI XML for an RDBS cache with all RDB schemas stored on the SINAI server.

```
Public Function LoadRDBS(ByVal inUserID As String, _
                         ByVal inSchemaID As String) As String
```

Returns SINAI XML for an RDBS cache with information on the relational DB schema identified by the inSchemaID argument.

```
Public Function SaveRDBSCache(ByVal inUserID As String, _
                              ByVal inCacheXML As String) As Boolean
```

inCacheXML is a SINAI XML representation of an RDBS cache, and the content of this cache will be stored in the default SINAI RDBS database on this server.

```
Public Function LoadAllSAMs(ByVal inUserID As String) As String
```

Returns SINAI XML for a SAM cache with all SAMs stored on the SINAI server.

```
Public Function LoadSAM(ByVal inUserID As String, _
                        ByVal inModelID As String) As String
```

Returns SINAI XML for a SAM cache with information on the SAM identified by the inModelID argument.

```
Public Function SaveSAMCache(ByVal inUserID As String, _
                             ByVal inCacheXML As String) As Boolean
```

inCacheXML is a SINAI XML representation of a SAM cache, and the content of this cache will be stored in the default SINAI SAM database on this server.

11. Concluding Remarks

Code generation from SIM and SAM can be useful to achieve more standardised architectures, at least for particular product lines, but the current SINAI prototype is primarily useful as a means to evaluate the SINAI approach. This because developing SINAI systems by hand is indeed very time-consuming, but using the prototype makes it possible to create at least prototypes and demo systems with much less effort. A practical evaluation of SINAI is also very important since SINAI originally set out to be a "pragmatic" project; i.e., its results should primarily be judged based on its ability to ease or improve the development of real software systems in a relatively short term.

The usefulness of automatically generated code should not be overestimated; e.g., complex business rules usually require non-trivial manual intervention. However, the code generated in SINAI, both the model-specific as well as the predefined code, is such that manual customisation, optimisation and specialisation is possible everywhere. Noticeably, the current prototype does *not* include support for preserving manual additions and changes when regenerating code, but such mechanisms can be included in our XML-based code generation approach without undue effort. The fact that all the code templates stems from the same XML template file makes it very easy to do manual additions to the generated code without involving the programs that does the actual generation. Thus this use of XML makes for a very flexible code-generation approach. It may be worthwhile to study this further in a separate project.

Main Deficiencies

The main deficiencies of the current prototype can be summarised as follows:

- Stored procedure generation is not quite ready. Only some minor coding and testing is missing, but it will also be necessary with a, preferably predefined, service to transfer SIM cache information to a database by invoking the stored procedures.
- It should be easier to refresh the content of a cache based on the content of another cache from the same SIM (this is possible, but a little awkward, in the currently generated caches).
- There should be more explicit support in the SIM caches for long, optimistic transactions.
- No IDL generation from SAMs
- No Rose model storage or generation from SAMs
- SIM modelling constructs like arrays and ordered collections are not currently supported (but can be added with very little effort).

As stated initially, and made clear by the description throughout this document, the current SINAI prototype is just that; a prototype primarily for "proof of concept". Much more effort will be needed to turn it into a full-scale, possibly commercial CASE tool. This may never be a goal for SINAI, however. If an overall SINAI evaluation turns out positive then the SINAI principles of evolutionary enterprise information system development will be its most important contribution.

12.References

1. E.P.Andersen; *SINAI - Concepts and Principles, A UML-based Architectural Framework for Evolutionary Information Systems*; NR Report from the SINAI project; http://www.nr.no/~egil/ sinai-concept-080601.pdf
2. E.P.Andersen; *Seamless Integration of Distributed Electronic Patient Records*; Deliverable WP2 D2.1 in SynEx. http://www.nr.no/~egil/ shs025-synex-client-wp2-d21.pdf
3. E.P.Andersen; *A Platform for Electronic Patient Record Integration*; Deliverable WP2 D2.2 in SynEx. http://www.nr.no/~egil/shs024-synex-server-wp2-d22.pdf
4. E.P.Andersen, B.E.Hansen, *Providing Persistent Objects to Globally Distributed Sites*, NOSA'99, 2nd Nordic Workshop on Software Architecture, University of Karlskrona/Ronneby, Research Report 13/99, 12-13.August 1999, Ronneby, Sweden; http://www.nr.no/~egil/brix-ws.pdf
5. E.P.Andersen; *Information Models for Component Design and Implementation*; ICSSEA'99, 12th Int'l Conf. on Software Systems Engineering and Applications December 8-10, 1999, Paris. http://www.nr.no/~egil/icssea99-im-comp.pdf , http://www.nr.no/~egil/icssea99-slides-im-comp.ppt
6. E.P.Andersen; *A UML-based Architectural Framework for Web-based Information Systems*; ICSSEA'2000, 13th Int'l Conf. on Software Systems Engineering and Applications December 8-10, 2000, Paris. http://www.nr.no/~egil/icssea00-sinai-slides.ppt
7. D.Box; *A Young Person's Guide to The Simple Object Access Protocol: SOAP Increases Interoperability Across Platforms and Languages*; http://msdn.microsoft.com/msdnmag/issues/0300/soap/soap.asp
8. BizTalk; http://www.biztalk.org
9. P.P.S.Chen; *The Entity-Relationship Model - Toward a Unified View of Data*; ACM Transactions on Database Systems, Vol.1, No.1, March 1976 p.9-36.
10. R.Elmasri, S.B.Navathe; *Fundamentals of Database Systems*; Addison-Wesley World Student Series 1989, ISBN 0-201-53090-2.
11. W.Grimson, D.Berry, J.Grimson, G.Stephens, E.Felton, P.Given, R.O'Moore, *Federated Healthcare Record Server - the Synapses Paradigm*, International Journal of Medical Informatics, 1998
12. P.Hurlen, K.Skifjeld, E.P.Andersen, *The Basic Principles of the Synapses Federated Healthcare Record Server*, International Journal of Medical Informatics, Vol. 52, Nr. 1-3, 1998
13. B.Jung, E.P.Andersen, J.Grimson; *Using XML for Seamless Integration of Distributed Electronic Patient Records*; XML Scandinavia 2000 Conference, Gothenburg, Sweden, May 2-4, 2000. http://www.nr.no/~egil/XML-2000-Scandinavia.pdf
14. B.Jung, E.P.Andersen, J.Grimson; *SynExML as a Vehicle for Electronic Patient Records*; XML Europe 2000 Conference, Paris, France, June 12-16, 2000. http://www.nr.no/~egil/XML-2000-Europe.pdf
15. H.T.Kung, J.T.Robinson, *On Optimistic Methods for Concurrency Control*, ACM Transactions on Database Systems, Vol.6, No.2, June 1981, p.213-226
16. Microsoft; (Distributed) *Component Object Model (COM)*; http://www.microsoft.com/com
17. Microsoft; *SOAP Toolkit 2.0 Gold Release*; http://msdn.microsoft.com/downloads/
18. Microsoft, *SQL Server*, http://www.microsoft.com/sql
19. Microsoft, *UDA/OLE DB/ADO*, http://www.microsoft.com/data
20. Microsoft, *IIS/ASP*, http://www.microsoft.com/iis
21. G.M.Nijssen, T.A.Halpin; *Conceptual Schema and Relational Database Design - A Fact Oriented Approach*; Prentice-Hall, 1989, ISBN 0-7248-0151-0 (newer versions are available)
22. OASIS; http://www.oasis-open.org
23. Object Management Group (OMG); *Model Driven Architecture*, http://www.omg.org/mda/

24. Object Management Group (OMG); *CORBA*, http://www.omg.org/corba
25. Object Management Group (OMG); *The Unified Modeling Language (UML)*; http://www.omg.org/uml
26. Rational Rose, *UML Resource Center*, http://www.rational.com/uml
27. *SOAP specification*, *SOAP specification*, http://msdn.microsoft.com/xml/general/soapspec.asp
28. Synapses Homepage, http://www.cs.tcd.ie/synapses/public/
29. SynEx Homepage, http://www.gesi.it/synex/
30. World Wide Web Consortium; *XML*; http://www.w3.org/XML
31. XML ORG; http://www.xml.org

Role Modelling

32. M.Aksit, L.Bergmans, S.Vural, *An Object-Oriented Language-Database Integration Model: The Composition-Filters Approach*, Proc.of ECOOP'92, European Conference on Object-Oriented Programming, Utrecht, The Netherlands, 1992, Springer-Verlag, LNCS 615, p.372-395
33. E.P.Andersen, *Conceptual Modeling of Objects - A Role Modeling Approach*, Dept.of Informatics, University of Oslo, ISBN 82-7368-176-9, November 1997
34. E.P.Andersen, T.Reenskaug; *System Design by Composing Structures of Interacting Objects*; Proc.of ECOOP '92, European Conference on Object-Oriented Programming, Springer-Verlag, LNCS 615, p.133-152
35. D.Bäumer, D.Riehle, W.Siberski, M.Wulf, *The Role Object Pattern*, Proc.of PLoP'97, the Conference on Pattern Languages and Programs, 1997; Technical Report WUCS-97-34, Washington University Dept.of Computer Science, 1997, paper 2.1, 11 pages.
36. M.Fowler, *Dealing with Roles*, Proc.of the 4th Annual Conference on the Pattern Languages of Programs, Monticello, Illinois, USA, September 2-5, 1997
37. E.Gamma, R.Helm, R.Johnson, J.Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994, ISBN 0-201-63361-2
38. G.Gottlob, M.Schrefl, B.Rock, *Extending Object-Oriented Systems with Roles*, ACM Transactions on Information Systems, Vol.14, No.3, July 1996
39. W.Harrison, H.Ossher, *Subject-Oriented Programming (A Critique of Pure Objects)*, Proc.of OOPSLA '93, Object-Oriented Programming Systems, Languages and Applications, ACM Sigplan Notices 28:10 1993, p.411-428
40. R.Helm, I.M.Holland, D.Gangopadhyay, *Contracts: Specifying Behavioural Compositions in Object-Oriented Systems*, Proc.of ECOOP/OOPSLA '90, European Conference on Object-Oriented Programming/Object-Oriented Programming Systems, Languages and Applications, ACM Sigplan Notices 25:10 1990, p.169-180.
41. I.M.Holland, *Specifying Reusable Components Using Contracts*, Proc.of ECOOP '92, European Conference on Object-Oriented Programming, Springer-Verlag, LNCS 615, p.287-307
42. R.E.Johnson, B.Foote, *Designing Reusable Classes*, Journal of Object-Oriented Programming (JOOP), June/July 1988, p.22-30+35
43. G.Kiczales, J.desRivières, D.G.Bobrow, *The Art of the Metaobject Protocol*, MIT Press, 1991
44. G.Kiczales, J.Lamping, A.Mendhekar, C.Maeda, C.V.Lopes, J-M.Loingtier, J.Irwin, *Aspect-Oriented Programming*, Proceedings of ECOOP'97, June 1997
45. E.A.Kendall, *Agent Roles and Role Models: New Abstractions for Multiagent System Analysis and Design*, Int'l Workshop on Intelligent Agents in Information and Process Management, German Conference on Artificial Intelligence, Bremen, Germany, September 1998, p.35-46
46. B.B.Kristensen, *Transverse Activities: Abstractions in Object-Oriented Programming*, Object Technologies for Advanced Software (eds: S.Nishio, A.Yonezawa), First JSSST Int.Symposium, Springer-Verlag, LNCS 742, p.264-278
47. B.B.Kristensen, *Transverse Classes and Objects in Object-Oriented Analysis, Design, and Implementation*, Journal of Object-Oriented Programming (JOOP), February 1993, p.43-51

48. B.B.Kristensen, *Object-Oriented Modeling with Roles,* Proc.of the 2'rd International Conference on Object-Oriented Information Systems, OOIS'95, Dublin, Ireland, 1995
49. B.B.Kristensen, K.Østerbye, *Roles: Conceptual Abstraction Theory and Practical Language Issues,* Theory and Practice of Object Systems, Vol.2, No.3, 1996, p.143-160
50. B.B.Kristensen, *Activities: Abstractions for Collective Behaviour,* Proc.of ECOOP '96, European Conference on Object-Oriented Programming, Springer-Verlag, LNCS 1098, p.472-501
51. B.B.Kristensen, J.Olsson, *Roles & Patterns in Analysis, Design and Implementation,* Proc.of the 3'rd International Conference on Object-Oriented Information Systems, OOIS'96, London, England, 1996
52. Paepcke, Andreas, *PCLOS: Stress Testing CLOS*, Proceedings of OOPSLA/ECOOP'90, Ottawa, Canada, October 1990, pp. 194-211
53. M.P.Papazoglou, *Roles: A Methodology for Representing Multifaceted Objects,* Proc.of DEXA '91, Database and Expert System Applications, 21-23.August, Berlin, Germany, Springer-Verlag, p.7-12
54. M.P.Papazoglou, B.J.Krämer, *A Database Model for Object Dynamics*, VLDB Journal, 6(2), 1997, p.73-96
55. B.Pernici, *Objects with Roles,* Proc. of ACM-IEEE Conference on Office Information Systems(COIS), Cambridge, Massachusetts, 1990
56. T.Reenskaug, P.Wold, O.A.Lehne, *Working with Objects, The OOram Software Engineering Method,* Manning/Prentice Hall 1995, ISBN 1-884777-10-4
57. T.Reenskaug , *Working with Objects: A Three-Model Architecture for the Analysis of Information Systems,* Journal of Object-Oriented Programming, May 1997, p.22-29+40
58. J.Richardson, P.Schwarz, *Aspects: Extending Objects to Support Multiple, Independent Roles,* Proc.of International Conference on Management of Data, May 1991, Denver, USA, ACM SigMod, Vol.20, No.2, June 1991, p.298-307
59. D.Riehle, *Composite Design Patterns,* Proc.of OOPSLA '97, Object-Oriented Programming Systems, Languages and Applications, ACM Sigplan Notices, 1997, p.218-228
60. D.Riehle, T.Gross, *Role Model based Framework Design and Integration,* Proc.of OOPSLA '98, Object-Oriented Programming Systems, Languages and Applications, ACM Sigplan Notices, 1998 (pages unknown)
61. J.Rumbaugh, *OO Myths: Assumptions from a Language View,* Journal of Object-Oriented Programming (JOOP), February 1997, p.5-7+48
62. R.J.Wirfs-Brock, B.Wilkerson; Object-Oriented Design: *A Responsibility-Driven Approach*; Proc.of OOPSLA '89, Object-Oriented Programming Systems, Languages and Applications, ACM Sigplan Notices 24:10, 1989, p.71-75.
63. R.J.Wirfs-Brock, B.Wilkerson, L.Wiener, *Designing Object-Oriented Software,* Prentice Hall, Englewood Cliffs, 1990
64. M.Van Hilst, D.Notkin, *Using Role Components to Implement Collaboration-Based Designs,* Proc.of OOPSLA '96, Object-Oriented Programming Systems, Languages and Applications, ACM Sigplan Notices, Vol.31, No.10, 1996, p.359-369
65. R.Wieringa, W.de Jonge, P.Spruit, *Roles and Dynamic Subclasses: A Modal Logic Approach,* Proc.of ECOOP '94, European Conference on Object-Oriented Programming, Springer-Verlag, LNCS 821, p.32-59
66. R.Wieringa, W.de Jonge, P.Spruit, *Using Dynamic Classes and Role Classes to Model Object Migration*, Theory and Practice of Object Systems (TAPOS), 1(1), 1995, p.61-83
67. L.Zhao, T.Foster, *Modelling Roles with Cascade,* to be published by the IEEE Software

A. ADO XML

The following is an example of ADO XML for a particular ADO recordset (in this a recordset in a SIMInfo cache that stores information about composite roles in SIMInfo).

Notice that the *<s:Schema>* part of the XML contains meta-information about the recordset, i.e., its fields, their types, and so on, while the *<rs:data>* part contains the actual object information.

```
<xml xmlns:s='uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'
      xmlns:dt='uuid:C2F41010-65B3-11d1-A29F-00AA00C14882'
      xmlns:rs='urn:schemas-microsoft-com:rowset'
      xmlns:z='#RowsetSchema'>
<s:Schema id='RowsetSchema'>
      <s:ElementType name='row' content='eltOnly' rs:updatable='true'>
            <s:AttributeType name='c0' rs:name='_ObjectID' rs:number='1'
                  rs:writeunknown='true' rs:basecatalog='SINAI-SystemDB'
                  rs:basetable='T_SIMINFO_CompositeRole' rs:basecolumn='_ObjectID'
                  rs:keycolumn='true'>
                 <s:datatype dt:type='uuid' dt:maxLength='16' rs:fixedlength='true'
                  rs:maybenull='false'/>
            </s:AttributeType>
            <s:AttributeType name='c1' rs:name='_TStamp' rs:number='2'
                  rs:rowver='true' rs:basecatalog='SINAI-SystemDB'
rs:basetable='T_SIMINFO_CompositeRole'
                  rs:basecolumn='_TStamp'>
                 <s:datatype dt:type='bin.hex' dt:maxLength='8' rs:fixedlength='true'
                  rs:maybenull='false'/>
            </s:AttributeType>
            <s:AttributeType name='IsAbstract' rs:number='3' rs:nullable='true'
                  rs:writeunknown='true' rs:basecatalog='SINAI-SystemDB'
                  rs:basetable='T_SIMINFO_CompositeRole' rs:basecolumn='IsAbstract'>
                 <s:datatype dt:type='boolean' dt:maxLength='2'
rs:fixedlength='true'/>
            </s:AttributeType>
            <s:AttributeType name='IsVirtual' rs:number='4' rs:nullable='true'
                  rs:writeunknown='true' rs:basecatalog='SINAI-SystemDB'
                  rs:basetable='T_SIMINFO_CompositeRole' rs:basecolumn='IsVirtual'>
                 <s:datatype dt:type='boolean' dt:maxLength='2'
rs:fixedlength='true'/>
            </s:AttributeType>
            <s:AttributeType name='RoleName' rs:number='5' rs:nullable='true'
                  rs:writeunknown='true' rs:basecatalog='SINAI-SystemDB'
                  rs:basetable='T_SIMINFO_CompositeRole' rs:basecolumn='RoleName'>
                 <s:datatype dt:type='string' rs:dbtype='str' dt:maxLength='80'/>
            </s:AttributeType>
            <s:AttributeType name='c5' rs:name='_SCode' rs:number='6'
                  rs:writeunknown='true' rs:basecatalog='SINAI-SystemDB'
                  rs:basetable='T_SIMINFO_CompositeRole' rs:basecolumn='_SCode'>
                 <s:datatype dt:type='i2' dt:maxLength='2' rs:precision='5'
                  rs:fixedlength='true' rs:maybenull='false'/>
            </s:AttributeType>
            <s:extends type='rs:rowbase'/>
      </s:ElementType>
</s:Schema>
<rs:data>
      <z:row c0='{220D7F43-5A58-11D5-A9A3-0060979B4844}' c1='
            0000000000000013e' IsAbstract='False' IsVirtual='False' RoleName='RoleA'
             c5='2'/>
      <z:row c0='{220D7F53-5A58-11D5-A9A3-0060979B4844}' c1='
            0000000000000013f' IsAbstract='False' IsVirtual='False' RoleName='RoleB'
             c5='2'/>
      <z:row c0='{E603CB0E-5B8B-11D5-A9A5-0060979B4844}' c1='
            000000000000016b' IsAbstract='False' IsVirtual='False' RoleName='BasicRole'
             c5='2'/>
      <z:row c0='{E603CAAC-5B8B-11D5-A9A5-0060979B4844}' c1='
            0000000000000162' IsAbstract='False' IsVirtual='True' RoleName='JointUnique'
             c5='2'/>
      <z:row c0='{E603CAAD-5B8B-11D5-A9A5-0060979B4844}' c1='
            0000000000000163' IsAbstract='True' IsVirtual='False' RoleName='Role'
             c5='2'/>
      <z:row c0='{E603CAB3-5B8B-11D5-A9A5-0060979B4844}' c1='
```

```
              0000000000000164' IsAbstract='False' IsVirtual='False' RoleName='ArrayIndex'
               c5='2'/>
       <z:row c0='{E603CABE-5B8B-11D5-A9A5-0060979B4844}' c1='
              0000000000000165' IsAbstract='False' IsVirtual='False'
RoleName='InformationModel'
               c5='2'/>
       <z:row c0='{E603CAC4-5B8B-11D5-A9A5-0060979B4844}' c1='
              0000000000000166' IsAbstract='False' IsVirtual='False'
RoleName='CompositeRole'
               c5='2'/>
       <z:row c0='{E603CACD-5B8B-11D5-A9A5-0060979B4844}' c1='
              0000000000000167' IsAbstract='True' IsVirtual='False'
RoleName='UniqueConstraint'
               c5='2'/>
       <z:row c0='{E603CAD3-5B8B-11D5-A9A5-0060979B4844}' c1='
              0000000000000168' IsAbstract='False' IsVirtual='False'
RoleName='Association'
               c5='2'/>
       <z:row c0='{E603CAEC-5B8B-11D5-A9A5-0060979B4844}' c1='
              0000000000000169' IsAbstract='False' IsVirtual='True' RoleName='LocalUnique'
               c5='2'/>
       <z:row c0='{E603CAED-5B8B-11D5-A9A5-0060979B4844}' c1='
              000000000000016a' IsAbstract='False' IsVirtual='False'
RoleName='ElementaryRole'
               c5='2'/>
</rs:data>
</xml>]
```

B. XML Templates for IDL, Visual Basic and SQL Schema generation

B.1 IDL XML Template

The following is the IDL XML template, used to generate IDL interface specifications for a particular SIM.

```
<?xml version="1.0"?>
<SINAI-IDL-Template>
    <InterfaceRoot> <!-- ==== START ======================================== -->
// SIIface<CacheName/>Cache.idl : IDL source for SINAI <CacheName/> Cache
//

// This file will be processed by the MIDL tool to
// produce the type library (SIIface<CacheName/>Cache.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";

[
       uuid(<GUID/>),
       version(1.0),
       helpstring("SINAI <CacheName/> Cache Interfaces v.1.0")
]
library SIIface<CacheName/>CacheLib
{
    // TLib : OLE Automation : {00020430-0000-0000-C000-000000000046}
       importlib("stdole32.tlb");
       importlib("stdole2.tlb");

       // Include the SINAI Basic Type Cache Interfaces
       // NB! Add path of this type library to "executable files" in
"Tools/Options/Directories"
       importlib("siifacebasictypecache.tlb");
       importlib("siifacecachemanager.tlb");

    // TLib : Microsoft ActiveX Data Objects 2.5 Library : {00000205-0000-0010-8000-
00AA006D2EA4}
       // Exists in catalog: C:\Program Files\Common Files\System\ADO
    // importlib("msado15.dll");

    // Forward declarations
    <InterfaceForward/>

    <InterfaceDefinition/>

};

//==================== EOF
    </InterfaceRoot> <!-- ==== END ========================================== -->

    <ISINAIInformationModel> <!-- ==== START ================================= -->
    //------------ ISINAIInformationModel

       [
              object,
              uuid(<GUID/>),
              oleautomation,
              dual,
              helpstring("ISINAIInformationModel interface"),
              pointer_default(unique)
       ]
       interface ISINAIInformationModel : IDispatch
       {
              // Cast to CacheManager interface
              [id(<UFID/>), helpstring("Function CastAsCacheManager")]
              HRESULT CastAsCacheManager([out, retval] ICacheManager**);

              // Get iterator for hver CR
        <GetCompositeRoleIterator/>

              // Get iterator for hver n'ary bi-assos
```

```
    <GetBiAssociationIterator/>

          // Transactional functions
          [id(<UFID/>), helpstring("Function BeginTransaction")]
          HRESULT BeginTransaction([out, retval] BSTR* transID);

          [id(<UFID/>), helpstring("Function CommitTransaction")]
          HRESULT CommitTransaction([in] BSTR transID, [out, retval] VARIANT_BOOL*);

          [id(<UFID/>), helpstring("Function RollbackTransaction")]
          HRESULT RollbackTransaction([in] BSTR transID, [out, retval] VARIANT_BOOL*);

          [id(<UFID/>), helpstring("Function GetNewGUID")]
          HRESULT GetNewGUID([out, retval] BSTR*);
   };
</ISINAIInformationModel> <!-- ==== END ================================= -->

<GetIterator> <!-- ==== START ============================================ -->
          [id(<UFID/>), helpstring("Function Get<EntityName/>It")]
          HRESULT Get<EntityName/>It([out, retval] I<EntityName/>**);
</GetIterator> <!-- ==== END ============================================== -->

<AbstractCompositeRoleInterface> <!-- ==== START ============================== -->
//------------ I<EntityName/>

   [
          object,
          uuid(<GUID/>),
          oleautomation,
          dual,
          helpstring("I<EntityName/> interface"),
          pointer_default(unique)
   ]
   interface I<EntityName/> : IDispatch
   {
       //------------ Cast functions

          [id(<UFID/>), helpstring("Function CastAsCommon")]
          HRESULT CastAsCommon([out, retval] ICommonIterator**);

          [id(<UFID/>), helpstring("Function CastAsRole")]
          HRESULT CastAsRole([out, retval] IRoleIterator**);

    <CastFunctions/>

       //------------ Creation functions

          [id(<UFID/>), helpstring("Function Update")]
          HRESULT Update();

       //------------ Attributes functions

    <AttributeFunctions/>

       //------------ Collections functions

    <CollectionFunctions/>

       //------------ Associations functions

    <RelationFunctions/>
   };
</AbstractCompositeRoleInterface> <!-- ==== END ============================== -->

<NonAbstractCompositeRoleInterface> <!-- ==== START ========================== -->
//------------ I<EntityName/>

   [
          object,
          uuid(<GUID/>),
          oleautomation,
          dual,
```

```
        helpstring("I<EntityName/> interface"),
        pointer_default(unique)
    ]
    interface I<EntityName/> : IDispatch
    {
        //------------ Cast functions

        [id(<UFID/>), helpstring("Function CastAsCommon")]
        HRESULT CastAsCommon([out, retval] ICommonIterator**);

        [id(<UFID/>), helpstring("Function CastAsRole")]
        HRESULT CastAsRole([out, retval] IRoleIterator**);

     <CastFunctions/>

        //------------ Creation functions

        [id(<UFID/>), helpstring("Function New")]
        HRESULT New([in] BSTR objectID);

        [id(<UFID/>), helpstring("Function Create")]
        HRESULT Create();

        [id(<UFID/>), helpstring("Function Update")]
        HRESULT Update();

        [id(<UFID/>), helpstring("Function Drop")]
        HRESULT Drop();

        //------------ Search functions

        [id(<UFID/>), helpstring("Function Find")]
        HRESULT Find([in] BSTR condition,
                     [out, retval] ICommonIterator**);

        [id(<UFID/>), helpstring("Function FindID1")]
        HRESULT FindID1([in] IRoleIterator* inObject,
                        [out, retval] ICommonIterator**);

        [id(<UFID/>), helpstring("Function FindID2")]
        HRESULT FindID2([in] BSTR inObjectID,
                        [out, retval] ICommonIterator**);

     <FindFunctions/>

        //------------ Attributes functions

     <AttributeFunctions/>

        //------------ Collections functions

     <CollectionFunctions/>

        //------------ Associations functions

     <RelationFunctions/>
    };
</NonAbstractCompositeRoleInterface> <!-- ==== END ========================== -->

<BiAssociationInterface> <!-- ==== START ============================== -->
//------------ I<EntityName/>

    [
        object,
        uuid(<GUID/>),
        oleautomation,
        dual,
        helpstring("I<EntityName/> interface"),
        pointer_default(unique)
    ]
    interface I<EntityName/> : IDispatch
    {
```

```
      //------------ Cast functions

         [id(<UFID/>), helpstring("Function CastAsCommon")]
         HRESULT CastAsCommon([out, retval] ICommonIterator**);

      //------------ Creation functions

         [id(<UFID/>), helpstring("Function New")]
         HRESULT New();

         [id(<UFID/>), helpstring("Function Create")]
         HRESULT Create();

         [id(<UFID/>), helpstring("Function Update")]
         HRESULT Update();

         [id(<UFID/>), helpstring("Function Drop")]
         HRESULT Drop();

      //------------ Search functions

         [id(<UFID/>), helpstring("Function Find")]
         HRESULT Find([in] BSTR condition,
                      [out, retval] ICommonIterator**);

    <FindFunctions/>

      //------------ Attributes functions

    <AttributeFunctions/>
  };
</BiAssociationInterface> <!-- ==== END ============================== -->

<DirAssociationInterface> <!-- ==== START ============================== -->
//------------ I<EntityName/>

  [
        object,
        uuid(<GUID/>),
        oleautomation,
        dual,
        helpstring("I<EntityName/> interface"),
        pointer_default(unique)
  ]
  interface I<EntityName/> : IDispatch
  {
      //------------ Cast functions

         [id(<UFID/>), helpstring("Function CastAsCommon")]
         HRESULT CastAsCommon([out, retval] ICommonIterator**);

      //------------ Creation functions

         [id(<UFID/>), helpstring("Function New")]
         HRESULT New();

         [id(<UFID/>), helpstring("Function Create")]
         HRESULT Create();

         [id(<UFID/>), helpstring("Function Update")]
         HRESULT Update();

         [id(<UFID/>), helpstring("Function Drop")]
         HRESULT Drop();

      //------------ Search functions

         [id(<UFID/>), helpstring("Function Find")]
         HRESULT Find([in] BSTR condition,
                      [out, retval] ICommonIterator**);

    <FindFunctions/>
```

```
      //------------ Attributes functions

    <AttributeFunctions/>
  };
</DirAssociationInterface> <!-- ==== END ============================== -->

<SpecificCastCompositeRole> <!-- ==== START ============================== -->
         [id(<UFID/>), helpstring("Function CastTo<EntityName/>")]
         HRESULT CastTo<EntityName/>([out, retval] I<EntityName/>**);
</SpecificCastCompositeRole> <!-- ==== END ============================== -->

<SpecificFindValue> <!-- ==== START ====================================== -->
         [id(<UFID/>), helpstring("Function Find<PropertyName/>")]
         HRESULT Find<PropertyName/>([in] <DataType/> inValue,
                                     [out, retval] ICommonIterator**);
</SpecificFindValue> <!-- ==== END ====================================== -->

<SpecificFindObject> <!-- ==== START ================================== -->
         [id(<UFID/>), helpstring("Function Find<PropertyName/>")]
         HRESULT Find<PropertyName/>([in] IRoleIterator* inObject,
                                     [out, retval] ICommonIterator**);
</SpecificFindObject> <!-- ==== END ================================== -->

<GetSetValue> <!-- ==== START ========================================== -->
         [id(<UFID/>), helpstring("Function Get<PropertyName/>")]
         HRESULT Get<PropertyName/>([out, retval] <DataType/>*);

         [id(<UFID/>), helpstring("Function Set<PropertyName/>")]
         HRESULT Set<PropertyName/>([in] <DataType/> inValue);
</GetSetValue> <!-- ==== END ========================================== -->

<GetSetObject> <!-- ==== START ========================================== -->
         [id(<UFID/>), helpstring("Function Get<PropertyName/>")]
         HRESULT Get<PropertyName/>([out, retval] I<InterfaceType/>**);

         [id(<UFID/>), helpstring("Function Set<PropertyName/>")]
         HRESULT Set<PropertyName/>([in] IRoleIterator* inObject);
</GetSetObject> <!-- ==== END ========================================== -->

<LoadSaveImage> <!-- ==== START ======================================== -->
         [id(<UFID/>), helpstring("Function Save<PropertyName/>")]
         HRESULT Save<PropertyName/>([in] BSTR inFileName);

         [id(<UFID/>), helpstring("Function Load<PropertyName/>")]
         HRESULT Load<PropertyName/>([in] BSTR inFileName);
</LoadSaveImage> <!-- ==== END ======================================== -->

<CollectionValue> <!-- ==== START ================================== -->
         [id(<UFID/>), helpstring("Function Insert<PropertyName/>")]
         HRESULT Insert<PropertyName/>([in] <DataType/> inValue);

         [id(<UFID/>), helpstring("Function Get<PropertyName/>")]
         HRESULT Get<PropertyName/>([out, retval] I<InterfaceType/>**);

         [id(<UFID/>), helpstring("Function Remove<PropertyName/>")]
         HRESULT Remove<PropertyName/>([in] <DataType/> inValue);

         [id(<UFID/>), helpstring("Function Count<PropertyName/>")]
         HRESULT Count<PropertyName/>([out, retval] long*);
</CollectionValue> <!-- ==== END ================================== -->

<CollectionObject> <!-- ==== START ================================== -->
         [id(<UFID/>), helpstring("Function Insert<PropertyName/>")]
         HRESULT Insert<PropertyName/>([in] IRoleIterator* inObject);

         [id(<UFID/>), helpstring("Function Get<PropertyName/>")]
         HRESULT Get<PropertyName/>([out, retval] I<InterfaceType/>**);

         [id(<UFID/>), helpstring("Function Remove<PropertyName/>")]
         HRESULT Remove<PropertyName/>([in] IRoleIterator* inObject);
```

```
            [id(<UFID/>), helpstring("Function Count<PropertyName/>")]
            HRESULT Count<PropertyName/>([out, retval] long*);
    </CollectionObject> <!-- ==== END ======================================== -->

    <GetDirAssociation> <!-- ==== START ====================================== -->
            [id(<UFID/>), helpstring("Function GetAssos<EntityName/>")]
            HRESULT GetAssos<EntityName/>([out, retval] I<EntityName/>**);
    </GetDirAssociation> <!-- ==== END ======================================== -->
</SINAI-IDL-Template>
```

B.2 Visual Basic XML Template

The following is the Visual Basic/ADO XML template, used to generate an ADO based Visual Basic implementation of a SIM cache for a particular SIM.

Too much code to be included in this report - see the "..\SINAI-SystemFiles\CodeTemplates\" directory.

B.3 SQL Schema XML Template

The following is the SQL Server schema XML template, used to generate an SQL Server schema for a particular SIM.

```
<?xml version="1.0"?>
<SINAI-RDBS-Template>
    <TableScript> <!-- ==== START ======================================== -->
--|*************************************************************************
--|
--| SINAI RELATIONAL DATABASE SCHEMA - TABLES
--|
--| This schema is automatically generated from the SINAI Information Model
--| <CacheName/>
--|
--|*************************************************************************

Set NoCount On

Print "Processing sinai-<CacheName/>-tables.sql:"

--|*************************************************************************
--| Initial Preparation
--|*************************************************************************

Print "Removing old foreign keys......"
    <DropForeignKeys/>
Print ".......finished removing foreign keys"

Print "Removing old views......"
    <DropViews/>
Print ".......finished removing old views"

Print "Removing old tables......"
    <DropTables/>
Print ".......finished removing old tables"


--|*************************************************************************
--| Tables
--|*************************************************************************

<CreateTables/>

--|*************************************************************************
--| PRIMARY KEYS
--|*************************************************************************

<CreatePrimaryKeys/>

--|*************************************************************************
--| FOREIGN KEYS
--|*************************************************************************

<CreateForeignKeys/>

--|*************************************************************************
--| INDEXES
--|*************************************************************************

<CreateIndexes/>

--|*************************************************************************
--| VIEWS
--|*************************************************************************

<CreateViews/>

-------------------------------------------------------------------------------

Print "Finished Processing sinai-<CacheName/>-tables.sql!"
```

```
Set NoCount Off

-------------- EOF ------------------------------------------------------------
    </TableScript> <!-- ==== END ================================== -->

    <BigSeparator> <!-- ==== START ========================================== -->
--------------------------------------------------------------------------------
--| <TitleText/>
--------------------------------------------------------------------------------
    </BigSeparator> <!-- ==== END ================================== -->

    <SmallSeparator> <!-- ==== START ========================================== -->
--| <TitleText/>
    </SmallSeparator> <!-- ==== END ================================== -->

    <DropTableStatement> <!-- ==== START ========================================== -->
If Exists (Select *
           From SysObjects
           Where ID = Object_Id('DBO.T_<ModelPrefix/>_<TableName/>'))
           Drop Table DBO.T_<ModelPrefix/>_<TableName/>
GO
    </DropTableStatement> <!-- ==== END ================================== -->

    <DropViewStatement> <!-- ==== START ========================================== -->
If Exists (Select *
           From SysObjects
           Where ID = Object_Id('DBO.V_<ModelPrefix/>_<ViewName/>'))
           Drop Table DBO.V_<ModelPrefix/>_<ViewName/>
GO
    </DropViewStatement> <!-- ==== END ================================== -->

    <DropForeignKeyStatement> <!-- ==== START ================================== -->
If Exists (Select *
           From SysObjects
           Where ID = Object_Id('<ModelPrefix/>_<ForeignKeyName/>'))
Alter Table T_<ModelPrefix/>_<ForeignKeySource/> Drop Constraint
<ModelPrefix/>_<ForeignKeyName/>
GO
    </DropForeignKeyStatement> <!-- ==== END ================================== -->

    <TableCompositeRole> <!-- ==== START ========================================== -->
--| TABLE T_<ModelPrefix/>_<TableName/>

Print "Installing Table T_<ModelPrefix/>_<TableName/>...."

Create Table DBO.T_<ModelPrefix/>_<TableName/> (
       _ObjectID      DTGUID        Not Null,
       _TStamp        TimeStamp     Not Null,
    <TableColumns/>
       _SCode         DTShort       Not Null Default 0)
GO

--------------------------------------------------------------------------------
    </TableCompositeRole> <!-- ==== END ================================== -->

    <TableNaryDirAssociation> <!-- ==== START ================================== -->
--| TABLE T_<ModelPrefix/>_<TableName/>

Print "Installing Table T_<ModelPrefix/>_<TableName/>...."

Create Table DBO.T_<ModelPrefix/>_<TableName/> (
       _AssosID     DTGUID      Not Null,
       _TStamp      TimeStamp   Not Null,
    _OwnerID     DTGUID      Not Null,
    <TableColumns/>
       _SCode       DTShort     Not Null Default 0)
GO

--------------------------------------------------------------------------------
    </TableNaryDirAssociation> <!-- ==== END ================================== -->

    <TableNaryBiAssociation> <!-- ==== START ================================== -->
```

```
--| TABLE T_<ModelPrefix/>_<TableName/>

Print "Installing Table T_<ModelPrefix/>_<TableName/>...."

Create Table DBO.T_<ModelPrefix/>_<TableName/> (
      _AssosID     DTGUID      Not Null,
      _TStamp      TimeStamp   Not Null,
    <TableColumns/>
      _SCode       DTShort     Not Null Default 0)
GO

-------------------------------------------------------------------------
    </TableNaryBiAssociation> <!-- ==== END ================================== -->

    <TableBinaryDirAssociation> <!-- ==== START ================================ -->
--| TABLE T_<ModelPrefix/>_<TableName/>

Print "Installing Table T_<ModelPrefix/>_<TableName/>...."

Create Table DBO.T_<ModelPrefix/>_<TableName/> (
    _OwnerID   DTGUID     Not Null,
    <TableColumns/>
      _SCode     DTShort    Not Null Default 0)
GO

-------------------------------------------------------------------------
    </TableBinaryDirAssociation> <!-- ==== END ================================== -->

    <TableBinaryBiAssociation> <!-- ==== START ======================== -->
--| TABLE T_<ModelPrefix/>_<TableName/>

Print "Installing Table T_<ModelPrefix/>_<TableName/>...."

Create Table DBO.T_<ModelPrefix/>_<TableName/> (
    <TableColumns/>
      _SCode    DTShort   Not Null Default 0)
GO

-------------------------------------------------------------------------
    </TableBinaryBiAssociation> <!-- ==== END ================================== -->

    <TableColumnStatement> <!-- ==== START ================================== -->
       <ColumnName/>    <ColumnDataType/>    <ColumnMandatory/> Null,
    </TableColumnStatement> <!-- ==== END ================================== -->

    <PrimaryKeyStatement> <!-- ==== START ======================================= -->
Alter Table T_<ModelPrefix/>_<PrimaryKeyTable/> Add Constraint
<ModelPrefix/>_<PrimaryKeyName/>
      Primary Key (<PrimaryKeyColumns/>)
GO
    </PrimaryKeyStatement> <!-- ==== END ================================== -->

    <ForeignKeyStatement> <!-- ==== START ======================================= -->
--|Alter Table T_<ModelPrefix/>_<ForeignKeySource/> Add Constraint
<ModelPrefix/>_<ForeignKeyName/>
--|      Foreign Key (<ForeignKeySourceColumns/>) References
T_<ModelPrefix/>_<ForeignKeyTarget/>(<ForeignKeyTargetColumns/>)
--|GO
    </ForeignKeyStatement> <!-- ==== END ================================== -->

    <CreateIndexStatement> <!-- ==== START ======================================= -->
Create <IndexIsUnique/> <IndexClustering/> Index <ModelPrefix/>_<IndexName/>
             On T_<ModelPrefix/>_<IndexTableName/>(<IndexColumns/>)
GO
    </CreateIndexStatement> <!-- ==== END ================================== -->

    <CreateViewStatement> <!-- ==== START ======================================= -->
--| VIEW V_<ModelPrefix/>_<ViewName/>

 Print "Installing View V_<ModelPrefix/>_<ViewName/>...."

Create View V_<ModelPrefix/>_<ViewName/>(<ViewColumns/>) As
```

```
  Select <ViewColumns/>
  From   <ViewTables/>
  Where  <ViewCondition/>
GO
------------------------------------------------------------------------------
    </CreateViewStatement> <!-- ==== END ================================== -->

    <ProcedureScript> <!-- ==== START ========================================== -->
--|*************************************************************************
--|
--| SINAI RELATIONAL DATABASE SCHEMA - STORED PROCEDURES
--|
--| This schema is automatically generated from the SINAI Information Model
--| <CacheName/>
--|
--|*************************************************************************

Set NoCount On

Print "Processing sinai-<CacheName/>-procedures.sql:"

--|*************************************************************************
--| Initial Preparation
--|*************************************************************************

Print "Removing old procedures......"

<DropProcedures/>

If Exists (Select *
           From   SysObjects
           Where  id = Object_ID('DBO.P_<ModelPrefix/>_DeleteAllInformation'))
   Drop Procedure DBO.P_<ModelPrefix/>_DeleteAllInformation
GO

If Exists (Select *
           From   SysObjects
           Where  id = Object_ID('DBO.P_<ModelPrefix/>_GetAllInformation'))
   Drop Procedure DBO.P_<ModelPrefix/>_GetAllInformation
GO

If Exists (Select *
           From   SysObjects
           Where  id = Object_ID('DBO.P_<ModelPrefix/>_GetModelSchema'))
   Drop Procedure DBO.P_<ModelPrefix/>_GetModelSchema
GO

Print ".......finished removing old procedures"

------------------------------------------------------------------------------

--|*************************************************************************
--| Create Procedures
--|*************************************************************************

<CreateProcedures/>

--|*************************************************************************
--| Special Retrieval Procedures
--|*************************************************************************

--| PROCEDURE: P_<ModelPrefix/>_DeleteAllInformation

Print "Installing P_<ModelPrefix/>_DeleteAllInformation...."
GO

Create Procedure P_<ModelPrefix/>_DeleteAllInformation As
Begin
    <SpecialDeletes/>
End --| P_<ModelPrefix/>_DeleteAllInformation
GO
```

```
Grant Execute On P_<ModelPrefix/>_DeleteAllInformation To Public
GO

-----------------------------------------------

--| PROCEDURE: P_<ModelPrefix/>_GetAllInformation

Print "Installing P_<ModelPrefix/>_GetAllInformation...."
GO

Create Procedure P_<ModelPrefix/>_GetAllInformation As
Begin
    <SpecialMetaSelects/>

    <SpecialSelects/>

/* ALTERNATIVE SOLUTION:
       Create Table <![CDATA[#]]>ResultInfo
       (__SIMrole__  VarChar(80),
        __Position__ Int)

       Insert Into <![CDATA[#]]>ResultInfo Values ("T_SIM_InformationModel", 1)
       Insert Into <![CDATA[#]]>ResultInfo Values ("T_SIM_InformationModel_Roles", 2)

       Select * from <![CDATA[#]]>ResultInfo
*/
End --| P_<ModelPrefix/>_GetAllInformation
GO

Grant Execute On P_<ModelPrefix/>_GetAllInformation To Public
GO

-----------------------------------------------

--| PROCEDURE: P_<ModelPrefix/>_GetModelSchema

Print "Installing P_<ModelPrefix/>_GetModelSchema...."
GO

Create Procedure P_<ModelPrefix/>_GetModelSchema As
Begin
    <SpecialMetaSelects2/>

    <SpecialEmptySelects/>
End --| P_<ModelPrefix/>_GetModelSchema
GO

Grant Execute On P_<ModelPrefix/>_GetModelSchema To Public
GO

-------------------------------------------------------------------------------

Print "Finished Processing sinai-<CacheName/>-procedures.sql!"

Set NoCount Off

-------------- EOF ------------------------------------------------------------
    </ProcedureScript> <!-- ==== END ================================== -->

    <SpecialDeleteStatement> <!-- ==== START ================================== -->
        Delete From T_<ModelPrefix/>_<TableName/>
    </SpecialDeleteStatement> <!-- ==== END ================================== -->

    <SpecialMetaSelectStatement> <!-- ==== START ============================ -->
        Select "<TableName/>" As __SIMrole__, <SequenceNumber/> As __Position__
    </SpecialMetaSelectStatement> <!-- ==== END ================================== -->

    <SpecialSelectStatement> <!-- ==== START ================================== -->
        Select * From T_<ModelPrefix/>_<TableName/>
    </SpecialSelectStatement> <!-- ==== END ================================== -->

    <SpecialEmptySelectStatement> <!-- ==== START ============================ -->
```

```
        Select * From T_<ModelPrefix/>_<TableName/> Where _SCode<![CDATA[<]]>0  --|
always return no rows
    </SpecialEmptySelectStatement> <!-- ==== END ================================ -->

    <DropProcedureStatement> <!-- ==== START ================================== -->
If Exists (Select *
           From SysObjects
           Where ID = Object_Id('DBO.P_<ModelPrefix/>_<ProcedureName/>'))
           Drop Procedure DBO.P_<ModelPrefix/>_<ProcedureName/>
GO
    </DropProcedureStatement> <!-- ==== END ================================== -->

    <CreateProcedureStatement> <!-- ==== START ============================== -->
--| PROCEDURE: P_<ModelPrefix/>_<ProcedureName/>

Print "Installing P_<ModelPrefix/>_<ProcedureName/>...."
GO

Create Procedure P_<ModelPrefix/>_<ProcedureName/>(<ProcedureArguments/>) As
    <ProcedureBody/>
End --| P_<ModelPrefix/>_<ProcedureName/>
GO

Grant Execute On P_<ModelPrefix/>_<ProcedureName/> To Public
GO

-----------------------------------------------
    </CreateProcedureStatement> <!-- ==== END ================================== -->

    <ProcedureArgumentStatement> <!-- ==== START ============================== -->
        @<ArgumentName/>  <ArgumentType/>  <ArgumentInOutCode/>
    </ProcedureArgumentStatement> <!-- ==== END ================================== -->

    <ProcedureCreateBody> <!-- ==== START ======================================== -->
    Begin
        Insert Into <TableName/>(<ColumnList/>)
                            Values (<ValueList/>)
    </ProcedureCreateBody> <!-- ==== END ================================== -->

    <ProcedureDropBody> <!-- ==== START ======================================== -->
    Begin
        Delete From <TableName/>
              Where <DropCondition/> And
                    (_TStamp <![CDATA[<]]>= @_TStamp)
    </ProcedureDropBody> <!-- ==== END ================================== -->

    <ProcedureBinaryDropBody> <!-- ==== START ============================== -->
    Begin
        Delete From <TableName/>
              Where <DropCondition/>
    </ProcedureBinaryDropBody> <!-- ==== END ================================== -->

    <ProcedureCRUpdateBody> <!-- ==== START ================================== -->
    Begin
        Update <TableName/>
           Set <SetList/>
         Where _ObjectID = @_ObjectID And
               _TStamp <![CDATA[<]]>= @_TStamp
    </ProcedureCRUpdateBody> <!-- ==== END ================================== -->

    <ProcedureNaryUpdateBody> <!-- ==== START ================================== -->
    Begin
        Update <TableName/>
           Set <SetList/>
         Where _AssosID = @_AssosID And
               _TStamp <![CDATA[<]]>= @_TStamp
    </ProcedureNaryUpdateBody> <!-- ==== END ================================== -->

    <ProcedureBinaryUpdateBody> <!-- ==== START ============================== -->
    Begin
        Update <TableName/>
           Set <SetList/>
```

```
          Where <UpdateCondition/>
      </ProcedureBinaryUpdateBody> <!-- ==== END ================================== -->
</SINAI-RDBS-Template>
```