A Multi-Agent Architecture for the Integration of Genomic Information

Ekard Burger, Johannes Link, Otto Ritter

German Cancer Research Center (DKFZ), Heidelberg
e-mail: {e.burger | j.link}@dkfz-heidelberg.de

We present an architecture of collaborating software agents which is supposed to tackle the problem of information integration in a flexible and generic way. An intended system is supposed to offer a transparent view onto information resources (data sources and software tools) for genomic research and to be able to adopt to the constantly evolving environment. This framework architecture is currently used for building a prototype of IGD-GIS (Integrated Genomic Database - Genome Information System).

1 Introduction

The HUMAN GENOME PROJECT (HGP) is one of the most important research efforts of our days. Its goal is to decipher the complete human genome till the year 2005. The genome represents the functional blueprint and evolutionary history of the human species.

A genome consists of a set of chromosomes, each chromosome being a sequence of DNA bases, which are at the computational level represented as the characters C, G, T and A. The first step in deciphering a genome is to determine these sequences (approximately $3*10^9$ bases in the case of the human genome). After sequence data has been generated, the subsequent step is to determine which regions within the sequence code for molecules that carry out a function within the organism. Functions include activities like transporting other molecules, catalysing biochemical reactions (e.g. enzymes) and adding structure to a cell.

Many of these functions are performed by proteins. The amino acid sequence of a protein is translated from a corresponding DNA sequence. However, only a small fraction of the whole genome is considered to specify the portions of our 50,000 to 100,000 genes that encode proteins. The rest of the DNA is partly important for controlling the mechanisms of protein production but mostly without known function.

The task of deciphering the genome goes far beyond mere sequencing but also includes acquisition and analysis of large amounts of additional information like genetic diseases and spatial structure.

Several primary sources of data have been set up for the HGP, e.g. for sequence information (EMBL, GENBANK), the GENOME DATABASE (GDB) for general gene-related information and a number of smaller databases for non-human genomes. The information about proteins is stored in sources like

PDB and SWISSPROT. Moreover, software tools for processing, combining, comparing and analysing the data and knowledge play a crucial role.

Thus, hundreds of heterogeneous and autonomous *information resources* (IRs) - i.e. databases **and** tools - exist and their number is still growing. Each of the IRs has its own interfaces and control languages, and represents information using conflicting data models and formats. The development of architectures and tools for effectively sharing and integrating relevant IRs is the prerequisite for the understanding of the information stored in the human genome.

We present in this paper a framework architecture for the intelligent integration of genomic (and other) information. The goal of the system is to hide the large variety of local or remote IRs and the disparity of their interfaces. A user agent (human or another system) should see only one interface and be able to query the system by specifying what it wants to know without a detailed knowledge where relevant information is located, what its representation is like and how the IRs' interfaces must be handled.

Crucial aspects which influenced the design of our *generic* architecture are adaptability and openness towards changes in its environment. We try to fulfill this goal by adopting the multi-agent paradigm and the explicitness of the system's metadata.

Contents Overview

Section 2 gives an overview about some existing approaches to handle the integration of information both in molecular biology and in general.

In section 3 we present some of the basic problems and our way to tackle them. Section 4 shows how these methods meet in our system architecture.

To demonstrate how the proposed architecture would work for the integration of different IRs, we give in section 5 a brief example how a sequence interpretation could be transformed and executed within our framework.

The final section summarizes the motivation for and the ideas behind our work.

2 Related Work

Database Integration

There exist several architectures and projects that address integration of information resources. Approaches from the database community are *data warehousing*, *multidatabase* and *federated database systems* [13]. These approaches do not scale very well to large set of autonomous resources. In particular, data warehousing is only feasible for a small set of sources, whereas the federated approach requires the development of an *integrated schema* for all combined sources.

Intelligent Information Integration

A widely appreciated approach is the INTELLIGENT INTEGRATION OF INFORMATION (I^3) architecture [22]. This ARPA sponsored project has developed a generic system architecture that defines a service layer to isolate applications from dynamic changes in the real-world environment. Modules in an I^3 system gather data from a variety of sources, and transform and process such data to

yield relevant information to end-user applications. I^3 modules manage metadata describing the content, format, location, and meaning of base data storage, manage consistency of data, its information content and its processing.

A concrete example for an implementation based on I^3 architecture is the TSIMMIS project [10]. TSIMMIS defines a dynamic, self-describing object model in which the integrated information is represented as a hierarchy of tuples containing identifier, label, type, and value for each object. Source-specific wrappers convert queries to native queries understood by the source and translate the answer from its native format back into the object model. The semantic transformations are described using a logic-based specification language.

A key problem with the integration of autonomous and heterogeneous information resources in an open environment like the internet is the constant change within IRs, of both their schemata and interfaces. The INFOSLEUTH project [2] has developed a system that retrieves and processes information in a continuously evolving environment. This agent-based architecture uses the concept of information brokerage to support a dynamic integration of available sources.

Information Integration in the Biomolecular Field

The most common approach to provide an integrated access to biomolecular information is hypertext navigation. The user can start in one DB to find a starting point, e.g. by searching for a gene in GDB. The result entry will then provide a set of links to related sources, e.g. to a sequence entry in EMBL.

Another popular way to integrate a set of heterogeneous sources is the data warehousing method. For each source to be integrated, we must define a translator from the source data format into the central warehouse data model. The converted data is later physically loaded into a single, homogeneous database. Queries can be applied to this warehouse database. An example of this approach is the IGD-I project [17]. This is only feasible for a limited set of sources and a more or less static environment.

Systems based on a virtual integration can avoid the mentioned problems. The BIOKLEISLI project [3] implemented a general-purpose query system, that provides a uniform query interface to a variety of information sources. The user can construct complex queries in the *Collection Programming Language*. A query explicitly identifies both the information sources that it applies to, and the tables and attributes that are to be queried within each source. The system optimizes and translates the query and submits the query to the responsible external source data driver. The results are returned in the common internal data exchange format. The transformation process is controlled with algebraic rules governing the use of the available IRs. Not having a unified global schema requires the user to have detailed knowledge about the data model of each IR.

GDB [6] or the DOCKING-D databases [1] avoid this problem. Both are based on the federated database approach. The user works only with a *federated schema* and the queries using this “world view” are dynamically translated into the single resources’ schemata. [12] and [15] contain an in-depth coverage of this topic.

3 Methods

In this section we describe a number of problems inherent to information integration, as well as the methods we suggest to tackle these problems.

Agents

Our architecture is based on the software agent paradigm [23]. The following requirements are best met within an agent-based environment:

- **Modularity**: New components can be added or removed without effecting the operation of other system parts.
- **Extensibility**: To allow new elements to be easily added to the system
- **Autonomy**: Agents can move between different locations.
- **Flexibility**: The ability to deal with the dynamic state of the system, e.g. availability of IRs or services.
- **Declarative forms of communication**: The communication between components in our system deal primarily with information and knowledge. Thus, communication protocols based on known agent communication languages [7], are more appropriate for our purpose.

Metainformation

In order to integrate information distributed among information resources of all kinds, we have to handle information about the resources themselves - called *metainformation* or *metaknowledge*.

In many previous integration projects, e.g. IGD-I [17], this metainformation was used in an implicit and procedural way. However, extensibility, flexibility and reusability of a system are crucially grounded in the explicitness of the system's underlying data and information. This metainformation must comprise knowledge about

- location and responsibilities of resources,
- their content and schemata,
- how to access and query them,
- the relations and dependencies between them.

Ontology

The problem of explicitly modelling knowledge and metaknowledge in order to be able to share and reuse it, has been tackled by various scientists in the field of Knowledge Engineering [8] [9]. They try to capture a domain's *ontology*, i.e. "an explicit specification of a conceptualization" [9]. Two well known languages for ontology specification are ONTOLINGUA [8] (a subset of KIF [7]) and CML [18].

In [20] a prospective ontology for Molecular Biology is introduced which tries to specify a consistent vocabulary taking into account the various meanings a word like "gene" can have in the domain.

Metabase

To be able to use and modify these ontologies, we have included in our architecture a database which is able to store the ontologies - i.e. the extended semantic schemata - of the metainformation as well as the metainformation itself. The fundamental difference between this *metabase* and a simple database is the concept of *metalayers*: Data on one layer can serve as a

schema for data on another layer. This approach is very similar to the ideas behind CONCEPTBASE, a deductive and object-oriented database [11].

Tasks

One main goal in our system architecture is to provide a flexible and open environment for the specification, execution and management of tasks. A task includes the description of problems and their respective solving strategy. The specification of a task is based on a *task description language* [5], which describes the goal, the problem solving strategy and the essential domain knowledge.

A flexible and modular programming environment should support the composition and integration of single tasks or modules into new combinations. The task descriptions form an essential part of the system's metabase.

Query Decomposition

When confronted with a query to our virtual information space a system faces two tasks:

1. Decide which of the external IRs are relevant to the query at hand based on the IR's contents description.
2. Decompose the original query into queries to the IRs.

We tackle this problem by adapted versions of the algorithms used in the INFORMATION MANIFOLD [14]. There, the contents of an IR is described by giving specifications for the available relations in the resource, as well as constraints about the range of these relations. The specifications are given in terms which are specified in a *world view*, i.e. a kind of unified domain ontology. The algorithms guarantee to find only relevant information resources.

Ontological Mapping

Having arrived at a set of queries to single IRs using our own domain ontology vocabulary faces us with another difficult problem: the transformation and mapping of the various data semantics for query and result transformation. Generic solutions for this problem - often addressed as *schema mapping* and *integration* - are ongoing research issues.

In order to be able to cope with new developments in the future we have chosen a two-step approach for the transformation task:

1. *Syntax transformation,* which is tackled by a grammar-based approach like in the ALCHEMIST project [21], serves to translate the various data formats into an internal object format.
2. *Ontological Mapping* translates information represented with commitment to one ontology into information committed to another ontology. This requires the specification of mapping rules as presented in [19].

Our two-step approach takes care that our system can easily adapt to future advances in the field of ontological mapping.

4 The IGD-GIS Architecture

Fig. 1 gives an overview of all the agents existing in our present architecture and their paths of communication. The flow of control and information within the system goes typically as follows:

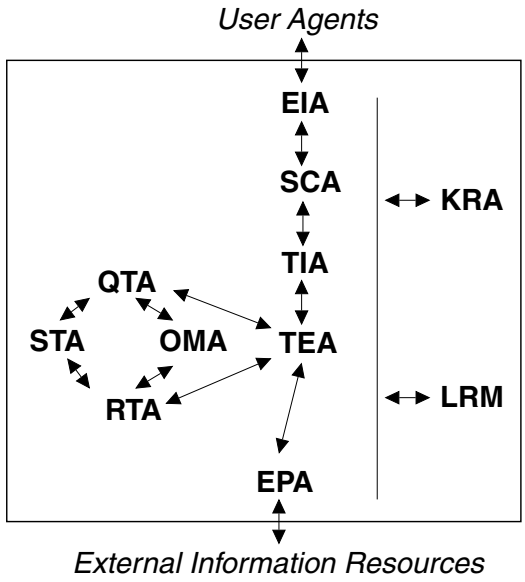

Fig. 1: Overview of system agents

A *user agent* posts a request for information retrieval, system update or reconfiguration to the *External Interface Agent* (EIA). This agent checks the user agent's authorization and decides if the request can be executed directly (e.g. a change of the system's repository) or if it has to deal with a complex retrieval action. In the latter case, the *Selector-Composer Agent* (SCA) takes over by decomposing the query into a set of subqueries to the single IRs together with appropriate composition operators for the query results. Afterwards it selects and parameterizes tasks which are able to handle these subqueries.

The *Task Invocation Agent* (TIA) expands a parameterized task into an executable *task script* by retrieving the task's specification and by expanding recursively all subtasks called in these specifications. This task script is passed over to the *Task Execution Agent* (TEA) for processing. The TEA coordinates the invocation of external IRs, the transformation of both queries to and results from the IRs, as well as the integration of the results. The integrated result is then handed back to the EIA - via TIA and SCA - which takes care of the retransformation into the appropriate form for the user agent.

Agent Capabilities

The system has a *weak* view of its agents, i.e. an agent in our system has no mentalistic notions like intentions, believes or desires. Each agent provides a set of capabilities for other agents, which we call the agent's services. It is autonomous in that it decides for itself which service it wants to provide at what time. Moreover, agents can migrate to different processors or networks.

Whereas the grouping of a set of services into one agent can partly be done arbitrarily, we tried to form groups which are likely to move and change together. It's possible, though, that during the evolution of the system an agent splits up, two agents melt or a service migrates from one agent to another, Besides, we can think of the same services being provided by lots of agents each of which uses different algorithms or different policies for service implementation.

Agent Cooperation

The system agents operate in a distributed environment, cooperating with other agents. Therefore, the agents require a mechanisms that allow them to communicate with each other. In our architecture all agent communication is managed by an *Agency*. The Agency is built using a CORBA framework [16] and takes care of dispatching messages to registered agents.

Agents use a subset of the KNOWLEDGE QUERY AND MANIPULATION LANGUAGE (KQML) [7] to communicate with each other, the KQML messages wrapping the service-dependent protocols in their contents slot. In order to invoke a certain service an agent does not specify the exact agent he wants to talk to, but broadcasts a '*request for service*' to a suitable group of agents. If more than a single agent is willing to provide a service, the service requester picks one (or maybe more) out of the group.

This communication approach uses *a service* as the finest granularity for agent negotiation. We have chosen it to enable a transparent communication which does not depend on the existence of certain agents but requires only that some agent can handle a certain service request.

Knowledge Repository

The *Knowledge Repository Agent* (KRA) is the frontend to the system's metabase which stores all important parts of configurable knowledge:

- The **System Repository** knows about agents, their services and status.
- The **Information Resources Repository** stores identity information, access information, local ontologies and contents descriptions of the external IRs.
- The **Task Repository** provides task specifications (scripts) and descriptions.

The point in having a central knowledge repository instead of knowledge distributed among the single agents is maintainability: all information which can change during a system's lifecycle is consistently held in one place.

Query and result transformation

The decomposition of queries into subqueries - performed by the SCA - which refer to a single IR can be done using the algorithms mentioned above. Thus, the *Query Transformation Agent* (QTA) only has to perform the translation from the internal query format and ontology into the a single IR's format and ontology. Just like in the *Result Transformation Agent* (RTA) a two-step approach is used:

- The *Syntax Transformation Agent* (STA) is called for grammar-based syntax transformation.
- The *Ontological Mapping Agent* (OMA) performs the transformation between ontologies on concept and instance level.

Whereas syntax transformation is a well-defined and manageable task, the problem of ontological mapping can by no means be regarded as solved. We suggest to integrate into our framework rule-based tools which are able to map terms of one ontology onto another.

External process invocation

The *External Process Agent* (EPA) provides services for accessing either basic protocols (ftp, http, mail, system calls) or specific database systems or maybe even some external IRs directly. Services take care of data format restructuring, i.e. they give their result in a *standardized stream format*.

There can exist many EPAs specialized for specific kinds of IRs, e.g. an EPA for relational databases and another for http connections.

Local Resource Management

Our *Local Resource Management Agent* (LRM) serves to give a transparent view of a single system with unified resources. It provides access to temporary data sources which must be shared among different agents. Agents accessing the LRM have to use the standardized stream format.

Adaptability of the Architecture

Our architecture provides adaptability and flexibility in several important ways:

- The integration of new agents providing additional or overlapping functionality is easy due to brokering and protocols. Thus we can imaging a system whose functionality evolves with new agents, e.g. an agent who automatically searches for new IRs to integrate.
- Agents can be located anywhere, since agency and LRM allow their transparent access.
- New IRs can be added and known IRs removed or temporarily disabled by just modifying the metabase. The system will recognize automatically which IRs are available at a certain time.

5 Example Scenario

To illustrate how the system is supposed to work internally when answering a query, we present a simple domain specific scenario and show how a user request is fulfilled.

Scenario

When biologists start to analyse a piece of sequence data, a very common procedure is to search for similar sequences in public databases. In our example a user is interested in all known human genes on chromosome X which have protein sequences similar to a given sequence (similarity is defined here by a "compare measure" being higher than 0.7). We imagine a user agent requesting the answer to a query in OQL (OBJECT QUERY LANGUAGE) [4]:

```
select struct(gene:x.name, protein:y.protein_id)
    from x in Gene, y in Protein
    where x.species = "human" and
    x.chromosome = "X" and
    x.protein_id = y.protein_id and
    ((y.sequence->compare("AATA...")) > 0.7)
```

The last part of the query applies a compare operation to a retrieved sequence object.

The domain ontology (world view) of the scenario - in terms of class hierarchy and additional relation is:

Class	Subclass	Attributes
Genome		Species
Chromosome	Genome	Chromosome
Gene	Chromosome	Gene_Name, Species, Chromosome, Sequence_id
Protein		Name, Sequence
Result		Result

World view relation:
compare (Result, Sequence, Sequence)

The capability description for each available IR is shown in Figure 2.

Query Decomposition

After checking the user's authorization the EIA decides that this request requires a complex retrieval action.

To answer a complex query involving several IRs, the first step is the transfor-

Source 1: Genome Database

Contents: V_1(Gene_Name, Species, Chromosome, Sequence_id) ⊆

Gene(c), GeneName(c, geneName), Species(c, species), Chromosome(c, chromosome), Sequence_id(c, seq_id)

Source 2: Protein Database

Contents:V_2(Protein_Name, Sequence) ⊆

Protein(p), Protein_name(p, seq_id), Sequence(p, Sequence)

Source 3: Compare tool for sequences

Contents:V_3(result, sequence_1, sequence_2) ⊆ compare(result, sequence_1, sequence_2).

Fig. 2: Contents specification

mation of the user query into our internal query format.This is done by the EIA:

q(Gene_Name,Sequence_id) ←
Gene(Gene_Name, **human**, **X**, Sequence_id),
Protein(Sequence_id, Sequence),
compare(result, Sequence, **AATA...**),
result > 0.7

Subsequently the SCA performs the query plan decomposition. A query plan is a sequence of accesses to IRs interspersed with local processing operations. A query plan consists of a set of conjunctive queries with additional information of the inputs and outputs to every subquery. The resulting query plan for our example:

Q_1: (V_1(c) ∧ Species(c, **human**) ∧ Chromosome(c, **X**)∧ Gene_Name(c,Gene_Name)∧ Sequence_id(c, Seqence_id))
Q_2: (V_2(p) ∧ Name(p, Sequence_id) ∧ Sequence(p,Sequence))
Q_3: (V_3(r, Sequence, "AATA...") ∧ r > 0.7)

Task selection and execution

Afterwards the following tasks are selected - because task type and resource fit - and parameterized by the SCA:

Input: Q_1, Q_2, Q_3
Output: set of (Gene_name, Sequence_id)
Subtasks:
Query_GDB,
Query_PIR,
Call_tool_blast,
Variables: v1, v2
BEGIN
JOIN (Query_GDB(Q_1,v1) Query_PIR(Q_2, v1,v2) Call_tool_blast(Q_3, v2))
END

The TIA extends the selected task description into an executable task script and invokes the TEA. The Query_GDB subtask would have the following expansion:

Taskname: Query_GDB
Task description: type(query), resource(GDB)
Input: Q_1
Output: set of (Gene_name, Sequence_id)
BEGIN
translate_query(Q_1, translated_query)
call_GDB (translated_query, result)
translate_result(result, return_res)
END

The TEA coordinates and supervises the execution of the task script. For our example we trace the execution of the Query_GDB task script. The first step for the TEA is the selection of a concrete agent, which can handle the translate_query task. In our system the QTA agent supports the translation from the internal query format to the concrete query format of the selected IR. The TEA calls the "translate query" service of the QTA with the following parameters:

translateQuery(Query, source syntax := internal syntax, source ontology:= worldViewOntology, destination syntax:= gdbOpmPerl, destination ontology:= gdbOntology)

The transformed query for the GDB OPM perl interface:

```
%x = (".db" => "hgd",
      ".class" => "Gene",
      ".command" => "query",
      "select" => [
          [ "attr" , "name" ],
          [ "attr" , "sequence_id" ],
               ],
      "where" =>
       [
          "match",
          [ "attr", "chromosome" ],
          "X"
       ]
     );
```

The transformed queries would then be given to the appropriate EPA(s), the returned results retranslated and joined.

The EIA would finally care for presenting the results in the representation required by the user agent.

6 Conclusion

We suggested in the preceding sections a new architecture for the integration of genomic (and other) information. None of the systems known to us are able to encompass **all** of the features at which our design aims:

- Transparent access to relevant data sources **and** software tools.
- Adaptability to a constantly changing environment, i.e. the change, appearance and disappearance of IRs can be coped with.
- Flexible and transparent distribution of functionality among agents and of agents over the net.

The architecture we have chosen tries to unify many of the ideas we have found useful in other approaches to intelligent information integration. The next step is to prove both the theoretical and technical feasibility of our framework. In order to achieve this, we are currently working on a prototype of IGD-GIS, the INTEGRATED GENOMIC DATABASE - GENOME INFORMATION SYSTEM. First results regarding usefulness and fulfilment of requirements will hopefully be available by the end of the year.

Acknowledgements

This work was partially funded by the European Community, BIOMED2 Workprogramme, contract BMH4-CT96-0263, and BIOTECH Workprogramme, contract BIO4-CT95-0037.

References

[1] **Aberer, K.**: The Use of Object-Oriented Data Models for Biomolecular Databases. *Proc. OOCNS 95 (Object-Oriented Computing in the Natural Sciences)*, Heidelberg, Germany, 1995.

[2] **Bayardo, R.; Bohrer, W.; Brice, R. et al.**: Semantic Integration of Information in Open and Dynamic Environments. *ACM SIGMOD Conference for Management of Data*, Tucsom 1997.

[3] **Buneman, P.; Davidson, S. B.; Hart, K.; Overton, C.; Wong, L.**: A Data Transformation System for Biological Data Sources. In *Proceedings of 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, September 1995.

[4] **Cattel, R.G.G. (ed.)**: The Object Database Standard: ODMG-93. *Morgan Kaufmann Publishers*, 1994.

[5] **Chandrasekaran, B.; Johnson, T.R.; Smith, J.W.**: Task-Structure Analysis for Knowledge Modeling. *Communications of the ACM*, vol. 35, no. 9, pp. 124-137, 1992.

[6] **Fasman, K.**: Restructuring the Genome Data Base: A Model for a Federation of Biological Databases. *Journal of Computational Biology*, vol. 1, no. 2, pp. 165-171, 1994.

[7] **Genesereth, M.R.; Ketchpel, S.P.**: Software Agents. *Communications of the ACM*, vol. 37, no. 7, pp. 48-53, 1994.

[8] **Gruber, Th. R.**: A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

[9] **Gruber, Th. R.**: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In Guarino, N. (Ed.): "*Formal Ontology in Conceptual Analysis and Knowledge Representation*". Kluwer Academic Publishers, 1993.

[10] **Hammer, J.; Garcia-Molina, H.; Ireland, K; Papakonstantinou, Y.; Ullman, J.and Widom, J.**: Information Translation, Mediation, and Mosaic-Based Browsing in the TSIMMIS System. In *Exhibits Program of the Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 483-487, San Jose, California, June 1995.

[11] **Jarke, M.; Gallersdörfer, R.; Jeusfeld, M. A.; Staudt, M.; Eherer, S.**: ConceptBase - a deductive object base for meta data management. In *Journal of Intelligent Information Systems*, Special Issue on Advances in Deductive Object-Oriented Databases, vol. 4, no. 2, pp. 167-192, 1995.

[12] **Karp, P.**: A strategy for database interoperation, *Journal of Computational Biology*, Vol. 2, No. 4, pp. 573-586, 1995.

[13] **Kim, W.**: Modern Database Systems. Addison Wesley 1995.

[14] **Levy, A.Y.; Srivastava, D.; Kirk, T.**: Data Model and Query Evaluation in Global Information Systems. *Journal of Intelligent Information Systems*, special Issue on Networked Information Discovery and Retrieval, 1995.

[15] **Markowitz, V. M.; Ritter, O.***:* Characterizing Heterogeneous Molecular Biology Database Systems. *Journal of Computational Biology*, vol. 2, no. 4, 1995.

[16] **Object Management Group, Inc.**: The Common Object Request Broker Architecture and Specification; Revision 2.0. Framingham, MA., July 1995.

[17] **Ritter, O**: The Integrated Genomic Database. In S. Suhai (Ed.) "*Computational Methods in Genome Research*", pp. 57-73. Plenum, New York 1994.

[18] **Schreiber, A. Th.; Wielinga, B.J.; Akkermans, J.M.; Van de Velde, W.; Anjewierden, A.**: CML: The CommonKADS conceptual modelling language. In L. Steels, A. Th. Schreiber, and W. Van de Velde (eds.) "*Proc. European Knowledge Acquisition Workshop EKAW'94*", Lecture Notes in Artificial Intelligence, vol. 867, pp. 1-25, September 1994.

[19] **Schreiber, A. Th.; Terpstra, P.; Magni, P.; van Velzen, M.**: Analysing and implementing VT using COMMON-KADS. In A. Th. Schreiber and W. P. Birmingham (eds.): "*Proceedings of the 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*", vol. 3, pp. 44-1 - 44-29, 1994.

[20] **Schulze-Kremer, S.**: Adding Semantics to Genome Databases: Towards an Ontology for Molecular Biology. *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology* (ISMB-97), to appear.

[21] **Tirri, H.; Lindén, G.**:. ALCHEMIST - an object-oriented tool to build transformations between Heterogeneous Data Representations. In Hesham-El-Rewini and Brice D. Shriver (eds.): "*Proc. of the 27th Annual Hawaii Int. Conf. on System Sciences*", vol. II, pp. 226 - 235, January 1994.

[22] **Wiederhold, G.**: Intelligent Integration of Information. Kluwer, 1996.

[23] **Wooldridge, M.; Jennings, N.R.**: Intelligent agents: theory and practice. *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152, 1995.