

The Semantic Interpreter Pattern

Pronab Ganguly, Fethi A. Rabhi* and Pradeep K. Ray

School of Information Systems, Technology and Management,

The University of New South Wales, Kensington,

NSW 2052, Australia,

**Tel: 61 2 9385 4179, Fax: 61 2 9662 4061; Email: f.rabhi@unsw.edu.au*

Keywords: Design Patterns, Ontology, Software Agents, Semantic Interoperation, Distributed Processing

1. Introduction

Software interoperability may be defined as the ability for multiple software components to interact regardless of their implementation programming language or hardware platform. The available mechanisms for software interoperability are [HOW,96]:

- Data-type interoperability: distributed and disparate programs support structured exchange of information through Application Programming Interfaces (APIs) invoked over a computer network.
- Specification-level interoperability: same as the previous one but also encapsulates knowledge representation differences at the level of abstract data types (e.g. a Table, Tree etc.). This enables programs to communicate at higher levels of abstraction and increases the degree of information hiding. CORBA and Enterprise Java Beans (EJB) fall into this category.
- Semantic interoperability: unlike the above two types of interoperability which are concerned with the form (structured description) at the integration interface, semantic interoperability represents design intent and predicted behavior as well as form (structured description) of the shared entities. It assumes that different information sources store information on related issues but each may offer a different meaning (semantics) of it.

2. The Semantic Interpreter Pattern

2.1 Context

The semantic interoperability problem may be defined in general as “the ability of a user to access, consistently and coherently, similar (though autonomously defined and managed) classes of digital objects and services distributed across heterogeneous

repositories, with federating or mediating software compensating for site-by-site variations”.

Thus semantic interoperability between various heterogeneous information sources continues to pose serious challenges to database, artificial intelligence and other related communities [OUK99]. This issue includes system, syntactic and structural/schematic interoperability. When we restrict the context to a scenario where various domain-specific but diverse databases are simultaneously accessed, semantic interoperability may be achieved. However, even in such cases, these databases are heterogeneous in the sense that they store different types of data, different data representation formats and there are different software and computing platforms used to run these databases. Researchers are working towards the objective of identifying the mechanisms that can access data from multiple databases without making changes to the existing databases.

2.2 *The problem*

Existing approaches for integration of heterogeneous databases include resolution of structural differences between underlying databases. In conceptual or global schema approach, there is a need for standardisation of data structures and definitions [RAM,96]. The conceptual schema specifies field and record definitions, structure and rules for updating data values. Various mappings and transformations are used to convert source data into a semantically equivalent and compatible form. The problems with this approach include:

- The emphasis is on schematic (i.e. syntactic) rather than semantic heterogeneity
- It assumes global knowledge is available which is not always possible
- The development costs of coding the system and the data definitions is huge
- The autonomy of individual databases is lost

In a federated database approach, a multidatabase language is used to facilitate interoperability. This language is the vehicle for interoperability. Additionally, each local database provides an export schema (a portion of its overall schema) which it is willing to share with other sources. Each database then uses these export schemas to define an import schema [LEC,99]. Thus a partial global schema representing information from other remote databases is generated and used. But the problem is that the users need to determine the meaning of all existing concepts and terms in every database across the network. This limits the number of databases and thus the scalability even in the same domain.

2.3 *Forces*

The major forces involved are:

- It is very difficult to represent semantic information outside a specific domain. Even in a specific domain, human (user) intelligence is usually required to be aware of the context so as to assess semantic information such as in a federated approach
- A specific action can have different results depending on the context. For example, in a telehealthcare application, the action "follow a normal diet" will mean different things to different users.
- When using a global schema, it is hard to maintain the autonomy of individual databases and keep the overall system's development costs as low as possible.
- Application-specific solutions that compromise the autonomy, flexibility and scalability of the entire distributed application should be avoided.

2.4 *Solution and Rationale*

Our solution is based on the concepts of *Ontologies* and *Ontological Agreements*. An ontology is an explicit specification of a body of formally represented knowledge including objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. The term is borrowed from philosophy and AI systems. We use this term in a much more pragmatic context where an ontology simply defines the vocabulary with which queries and assertions are exchanged amongst a system of interacting entities. Ontological agreements are commitments to use the shared vocabulary in a coherent and consistent manner. The communicating entities sharing a vocabulary need not share a knowledge base; each knows things the other does not, and a communicating entity that commits to an ontology is not required to answer all queries that can be formulated in the shared vocabulary.

Our proposed *Semantic Interpreter Pattern* addresses the forces as follows:

- Since contexts need to be used to represent the underlying semantics of the databases, these contexts are defined in a *domain ontology*. The terms of the ontology are linked with the corresponding databases or repositories through *ontological agreements*. So the users only need to express their needs by using terms in the ontology.
- There are *software agents* collaborating with every (human) user in the distributed environment as well as the ontology and the databases. These software agents are considered like human entities (having beliefs, desire and commitments) constrained by pragmatics (intentions, communication etc) as well as semantics (meanings, propositions, validity etc) and syntax (formal structure, data etc). They handle all communication aspects, receive feedback from the user, initiate communication, monitor events and perform certain tasks .
- These agents can manage changes by collaborating with other agents. The pattern achieves semantic interoperation, in a specified domain, through a multi-agent system.

2.5 Structure diagram

The diagram in figure 1 illustrates the structure of our pattern. It consists of following elements:

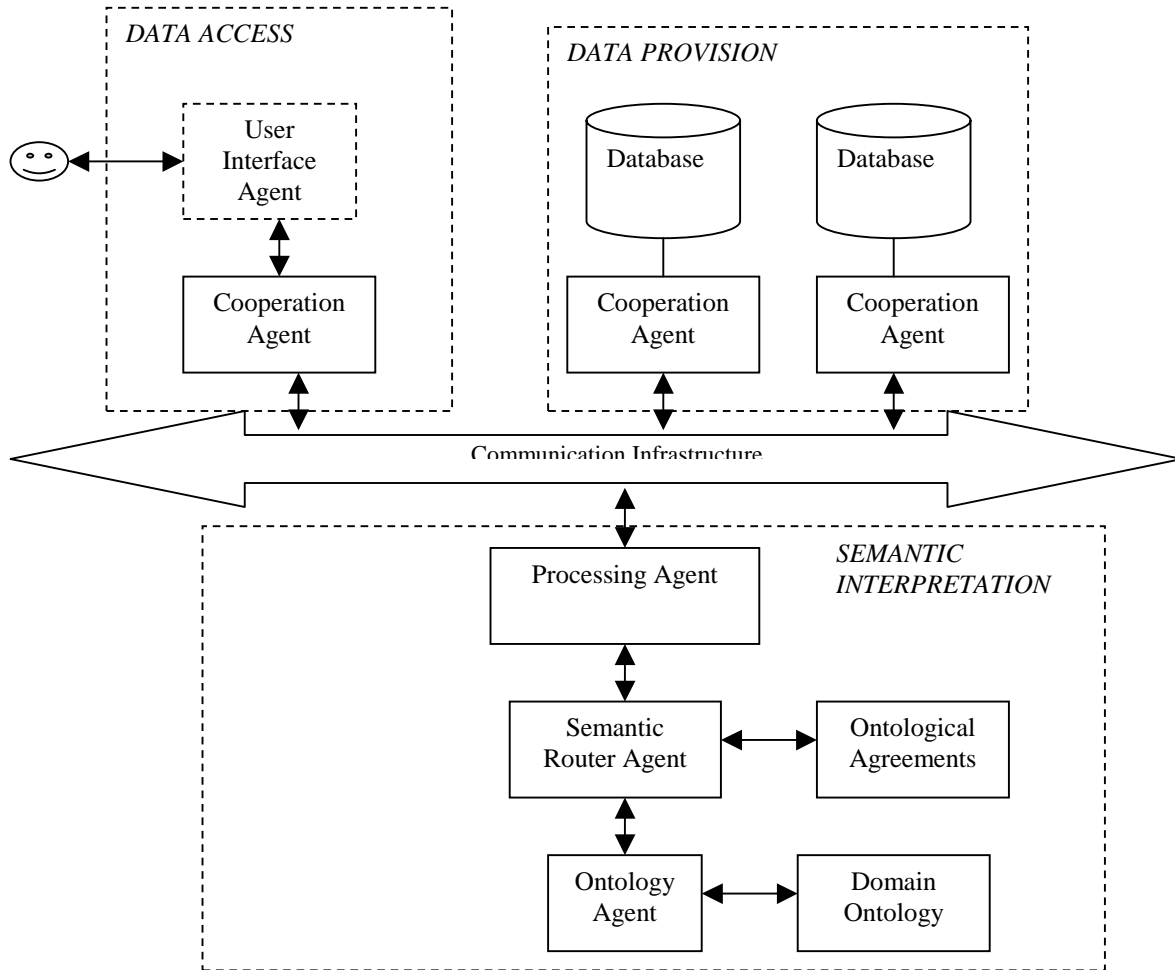


Figure 1: Structure of a software agent based system for semantic interpretation

2.5.1 Cooperation Agent:

It coordinates high level query processing. It manages information about local data source or users. It processes queries initiated by itself or other Cooperation Agents.

2.5.2 Ontology Agent

It manages the ontology which describes the domain-specific concepts and terms. It responds to requests from the Semantic Router Agent.

2.5.3 Semantic Router Agent

It stores the ontological agreements. It helps the Processing Agent to interpret the meaning of certain events using these ontological agreements.

2.5.4 Processing Agent

It processes the request from another Cooperation Agent in terms of ontological information. It may use the Semantic Router Agent to identify the relevance of an event and defines execution plans.

2.5.5 User Interface Agent

This is an interface agent between the user and a Cooperation Agent. Each user interacts with only one Cooperation Agent even though it might require the help of other Cooperating Agents to access the information.

2.6 *Interaction diagram*

Figure 2 shows an example of interactions between the different components the pattern. Section 2.8 will describe some scenarios within a particular application domain.

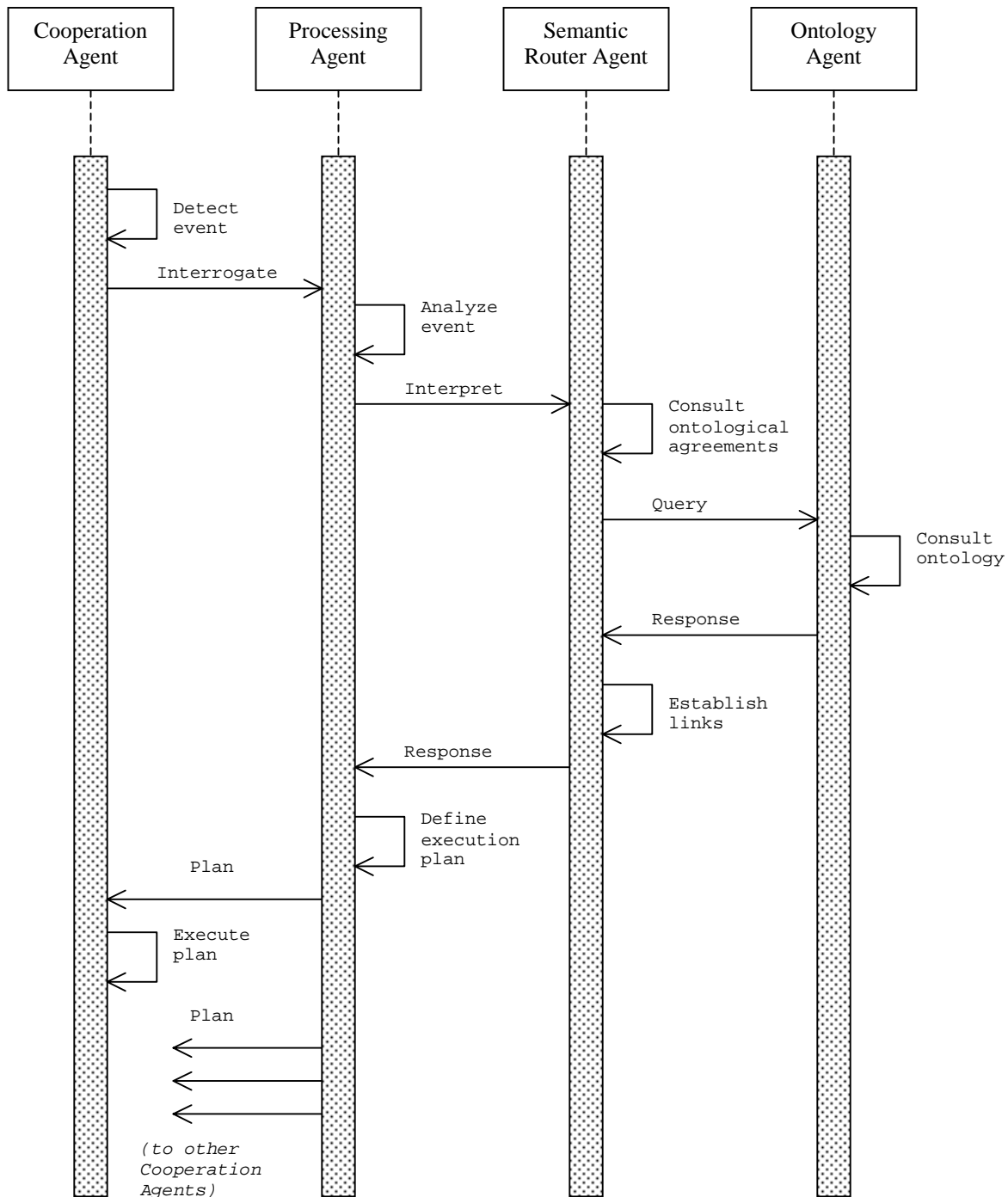


Figure 2: Example of interactions between the different components of the pattern

2.7 Implementation issues

The implementation of our pattern relies on a software agent cooperation framework. Approaches to define such frameworks include Agent Communication Languages (ACL). Though KQML is a widely known ACL, FIPA (Foundations of Intelligent Physical

Agents) have specified another ACL [TAH,00]. FIPA-ACL supports the concept of *communicative acts* which directly map into the communication primitives associated with our agents. Examples of communicative acts include “query”, “request”, “inform” and “failure”. FIPA-ACL is supported by a visual tool called IPEditor [TOS,00] for the development of multi-agent systems.

2.8 *Known Uses*

2.8.1 Telehealthcare

Our main use of the Semantic Interpreter Pattern is in Telehealthcare where it is under implementation. The focus is on diabetes management. The role of a User Interface Agent includes:

- Managing a patient's diet control, exercise routine, etc.
- Monitoring blood glucose readings (through a Glucometer which may be connected to a PC)
- Giving advice about oral medication/ insulin injections

As an example, there is a change in the environment of the patient such as the replacement of the Glucometer by another one that uses different measurement units (e.g. mmol/l to mg/dl). In this case, the Cooperation Agent detects an abnormal change in the data sampled and communicates this event to the Processing Agent. The Processing Agent queries the Semantic Router Agent which first obtains the ontological agreements related to this event and then collaborates with the Ontology Agent to access the ontology. Subsequently, the Processing Agent identifies the relevant information sources and creates an execution plan which is sent back to respective Cooperation Agent/s. The execution plan could be to make a unit conversion when the data is obtained, or change data contained in databases according to the new unit or keep both units stored at the same time (the Data Access Cooperation Agent would have to distinguish between old and new data).

In a more complex case, suppose that a "Normal diet" is recommended for the patient. Depending on the location and background, this will mean different combinations of foods to different patients. The ontology will specify a representational vocabulary for the above scenarios in an appropriate context. When the Cooperation Agent detects any change in the background details of user (patient identity, location etc), it notifies the Processing Agent which, with the help of the Semantic Router Agent and the Ontology Agent interpret the semantic context and presents the interpretation to the user [GAN,00], [RAM,96].

2.8.2 Other application areas

Other application domain includes integration of heterogeneous geographic information systems. The Interoperable Geographic Information System (GIS) adopts [LEC,99]

context mediation approach. In this approach, the emphasis is not on static integration of export schema but on dynamic resolution of semantic conflicts. The context comparison is carried out by collaboration between Cooperation Agent, Ontology Agent, Semantic Router Agent and Processing Agent. This collaboration enables users to access remote databases without knowing their semantics or representations.

2.9 *Related Patterns*

This pattern is similar to patterns for federated architectures with the additional benefit of semantic interoperation. The following agent based federated architectures patterns have been identified [FER00]:

- The Federation Pattern: this pattern reflects inter-domain dependencies.
- The Dependency Separation Pattern: this pattern reflects inter-application dependencies.
- Interface connection pattern: this pattern reflects domain based communication mechanisms.

The above patterns are derived from Alexander's pattern form [ALE,79]. However none of the above patterns explicitly addresses the structural aspect for semantic interoperation. Our pattern addresses the structural aspect of semantic interoperation explicitly and relies on an agent framework pattern adapted from FIPA ACL primitives [TAH,00] as well as an ontology.

3. Conclusion

A domain specific design pattern for semantically interoperable systems is proposed. It is based on the idea of software agents and software mediated ontological agreements. However it is a serious challenge to develop and maintain a comprehensive and ideal ontology which will have clarity, coherence, extensibility etc. In that case there will be a requirement for semantic conflict resolution and further research is being conducted to address this aspect. More research is also required into issues such as scalability, crash recovery, inconsistencies and security in order to use software agent technology more effectively [GAN,00].

4. References

[ALE,79] Alexander C et al; A Pattern Language, Oxford University Press, New York, 1979

[FER,00] Fernandez George & Zhao Liping; A Pattern Language for a Federated Architecture; Proceedings of KoalaPloP 2000; 24-26 May; Melbourne Australia, Technical Report TR-00-7, Department of Computer Science, RMIT, Australia, pp. 21-31.

- [GAN,00] Ganguly P, Mazzi C & Ray P, Software Agent Based Personal Assistant for Tele-Homecare in Diabetes Treatment; Proceedings of MAMA 2000 (Paper #1574-367); December 11-15, 2000, Wollongong, Australia
- [HOW,96] Howie T C, Kunz J C, Law K H; Software Interoperability; Center of Integrated Facility Engineering; Stanford University; 12/25/1996.
- [LEC,99] Leclercq E, Benslimane D & Yetongnon K; ISIS: A Semantic Mediation Model and an Agent Based Architecture for GIS Interoperability; Proceedings of International Database Engineering & Applications Symposium; 1999; <http://church.computer.org/proceedings/ideas/0265/026500087.pdf> accessed on 29/12/99
- [OUK,99] Ouksel M Aris; A Framework for Scalable Agent Architecture of Cooperating Heterogenous Knowledge Sources; Intelligent Information Agents; Matthias Klush (Ed.); Springer; 1999; page 100 - 124.
- [RAM,96] Ramesh v, Canfield Kip, Quirologico Steve & Silva Marcelo; An Intelligent Agent-based Architecture for Interoperability among Heterogeneous Medical Databases; <http://hsb.baylor.edu/ramsower/ais.ac.96/papers/ramesh2.htm> accessed on 12/23/1999
- [TAH,00] Tahara Yasuyuki, Ohsuga Akhiko & Honiden Shinichi; Agent Communication Language Patterns and Their Tool Support; Proceedings of MAMA 2000 (Paper #1574 351); December 11-15, 2000, Wollongong, Australia
- [TOS,00] Toshiba Corp; Bee-gent WWW page <https://www2.toshiba.co.jp/beegent/>, accessed on 26/12/00