

Ontology fusion approach for integration in heterogeneous distributed systems

Abdulrab, H ², abdulrab@insa-rouen.fr
Babkin E. ³, babkin@unc.sci-nnov.ru
Kazakov M. ^{1,2}, m-kazakov@eads.matradatavision.com

¹ Open Cascade S.A. R&D, (An EADS Matra Datavision company), 31, avenue de la Baltique, 91954 Les Ulis CEDEX, France, Tel : +33 1 69 82 28 79.

² INSA de Rouen, BP 08, pl. Emile Blondel, 76131, Mont Saint Aignan, France

³ Nizhny Novgorod State Technical University, Applied Mathematics department, 24, Minin street, Nizhny Novgorod, 603024, Russia.

Abstract

In this article we sketch an approach which aims a developer to reduce difficulties in integration and adaptation of software entities into a heterogeneous distributed system. It is achieved by separate implementation of entities and supplying them with logical descriptions in form of facts and axioms. According to our approach, a software entity consists of procedural and declarative parts. Procedural part can be concerned as a number of software patterns aiming meta-implementation of interaction strategies and declarative part which is the coarse-grained knowledge of entity preferences with relevant external information. Knowledge is formally expressed in form of ontology and axiomatic logical model. The declarative part together with reflection capabilities are used for deriving the specialization of varying parts of the entity patterns at runtime. Querying mechanisms is used to initiate the execution process.

Introduction

Undoubtedly the key features of structure and behavior of modern enterprise and industry systems should be reflected by models and architectures of modern software. Thus a successful software system should become distributed and autonomous.

In a distributed system that consists of a large number of independent software entities it is practically required to apply some Common Integration Methodology (CIM) in order to force these entities working together in a coordinated way and share their data. Under the term “software entity” we consider any software program presumed to work in the distributed environment by communication with other entities. We suppose that from a general point of view a CIM should mainly resolve the following issues:

- Knowledge integration - We use generic definition of knowledge term as it was given initially in [TUR01]. According to this source knowledge can be recognized as a set of different data resources that allows prediction of future states of world model. Thus this term covers all information that allows a software system to establish rational actions in order to satisfy properly its own goals. More pragmatically, knowledge is a specification of some concepts, their attributes and their interrelations. Knowledge integration is represented as the process of consolidation of separate partial knowledge that leads to entity ability to produce conclusions about concepts, attributes and interrelations of other entities and in practice to deal with varying and unfair data structures.
- Behavior coordination - In cybernetics behavior is concerned usually as a protocol of an observed system's changes from one state to another. Behavior of a software entity consists of a set of actions that can be performed. The emergent moving from strict client-server architectures to architectures with loosely coupled components requires paying great attention to algorithms of actions coordination among multiple uncorrelated entities. Thus behavior coordination can be concerned as a dynamic composition of entities algorithms resulting in development of coherent execution environment where every entity is aware of execution state and operational semantics of any other related entity.

From a software architect point of view integration of entities in complex distributed environments faces usually many design and development problems. These are: protocols compatibility, absence of the API of the needed service on time of compilation, problems of configuration and dynamic deployment and many others. Besides there is a set of problems, related to the absence of certain information about entities of distributed system in runtime. For example, we consider someone replacing a component in a working system with its new version without shutting down the entire environment. The new component, intended to perform the same functionality, has a little bit different API. This may be followed, for instance, by the more powerful functionality of this component, which requires several optional functions. However, the client of the system doesn't know anything about additional parameters, thus can not use anymore the component. Another example is an adaptation for

dynamically changing complex configuration of environment (communication protocols, signatures of methods and other information).

Above mentioned problems led us to work on the partial description of software model by a formal knowledge description methodology. This helps the integration process by adding semantic information to a distributed heterogeneous environment. The mechanism to integrate components by their semantic descriptions may lead to reduce the amount of manual work to implement the systems. Some precise information, which can't be available at the steps of compilation or configuration, may be described using ontologies for further dynamic reconfiguration. Communication of systems by protocols carrying one semantic meaning, but complying with different standards, is an example of such kind of integration.

For different modern distributed software architectures the issues given above have different forms and there are different points of interest inside these issues. We can mention briefly two different architectures and point out the specific problems in development of CIM for these architectures:

- Distributed objects: CORBA paradigm, proposed by the OMG. CORBA provides a great level of interoperability and huge possibility of distribution, but Interface Description Language (IDL) [IDL01], used in CORBA doesn't allow full semantic description of distributed components and objects.
- Mobile Agents paradigm [WOO01] has grown during the last decades and provides the conception of distributed, autonomous entities, but currently there are no "working" standards and integration of mobile agent systems and that is a complex problem in this technology.

None of the mentioned above architecture (also JINI, EJB) provides the full integration of component semantic and that leads to wasting time for systems development and maintenance and reduce the reliability and flexibility of system. That is why in this work we are interested in the research of a common integration model (CIM) applicable for Mobile Agents and CORBA systems.

Integration via Ontology fusion

General considerations

We developed the approach to knowledge integration and behavior coordination using combination of the concept of software patterns and ontological descriptions inside each component. This approach was named "Ontology Fusion". In our approach entity's behavior is implemented in the procedural form as a software pattern (or a group of patterns) in form of a simple state machine. A software pattern as it was introduced in [GAM01] describes "well-known solution of well-known problem" in form of UML specification. The Model-View-Controller abstraction can be presented also as such a pattern. Besides, each entity is shipped by the ontology that plays two roles –declares partial semantic world and defines variable parts of the pattern used. Thus the software entity consists of tightly linked procedural and declarative parts.

In order to share knowledge and build the united semantic model inside the whole distributed software system entities exchange their ontologies on the basis of preferable communication protocols. Then, various formal logical approaches for ontologies manipulation and querying are used in order to adapt entity's software patterns for united semantic model.

In our opinion the important added value of this approach is that during ontology manipulation and modification of an entity, its software pattern is logically correctly, independently and automatically transformed into a specialized form according to the current environment and tasks performed. This feature allows performing knowledge integration and behavior coordination by formal manner for multiple independent software entities at the same time. In comparison with other approaches of loosely coupled software integration (Message Oriented Middleware [MOM01]), ontology fusion proposes the powerful, logically verified runtime integration of software entities, based on their semantic definition. It allows to preserve structure consistency, reduce the time of reconfiguration of complex heterogeneous environments and to reduce the amount of "hardcoded" information, which can be resolved in runtime from the semantic definition. The important point of our approach, is that software entities, carrying ontologies, may "talk" (perform the fusion process), defining APIs, protocols, parameters in a complex heterogeneous environment without paying many attention to the aspect of incompatibility.

On the other hand, our approach requires additional computation time for the ontology fusion process and reflection techniques, based on the analysis of the ontology fusion results. This time is more, than direct "hardcoded" connection or a pure reflection technique usage. Another disadvantage of our approach is the complexity of logical declarative languages for description of ontologies and their interrelations for the use by software developers. Also it is necessary to mention, that our solution requires at least some preliminary platform interoperability, such as using the same network protocols, same mechanism to perform ontology fusion and the same language for ontology definition.

Concept and implementation

Ontology fusion in our approach is a multistage process during which the integration of knowledge and behavior is performed. We distinguish three main stages in Ontology Fusion process:

- Ontology alignment
- Logical inference
- Weaving process

In order to explicate ontology and to perform consequent ontology alignment and logical inference we have investigated suitability of the formal approach based on Frame-logic (F-logic) formalism [KIF01] (we use SiLRI [DEC01, ONT01] inference engine, working with F-Logic). F-Logic offers a clean declarative fashion to define classes, their instances, inheritance and methods (called relations in F-Logic terms). It allows the declaration of logical clauses (axioms) that define the different relations and constraints among different variables and methods. It's necessary to mention, that acquisition of knowledge and construction of ontologies are the tasks out of the scope of this article.

Recently the DAML+OIL [DAM02] language was developed. This language is based on excellent research results in Description logics and use *SHIQ* logic as its basis [HOR01]. *SHIQ* logics is decidable [JEF01], however its expressiveness is less than corresponded F-Logic parameter. During our research we're experimenting with different expressiveness with leak of decidability. It's sure, that in real applications decidable logic shall be used. Additional argument using the F-Logic is that its theorem prover is written in java and this makes integration and prototyping easier.

The concept of Ontology alignment is well described in [NOY01]. To align ontologies we use very simple exact coincidence of top level names and attributes of objects in our ontologies. It is done implicitly during loading facts, axioms and queries inside Ontology Fusion engine. The definition of ontology alignment rules is also specified by axioms that define some relations and bindings among some variables of two ontologies. The consequent logical inference can be seen as a logical resolution of queries that are applied to the aligned ontologies.

The last stage of Ontology Fusion presumes the use of results of inference either for precision of data or for weaving process. Weaving process uses reflective capabilities of an entity's programming languages to call certainly defined methods and apply defined variables values that were deduced from previous stages of fusion.

The detailed process of knowledge and behavior integration and relation with different issues of agent functionality is shown in the Figure 1 in terms of generic agent-oriented system. Here the conversation pattern [BAK01] is applied for definition of agent's interactions.

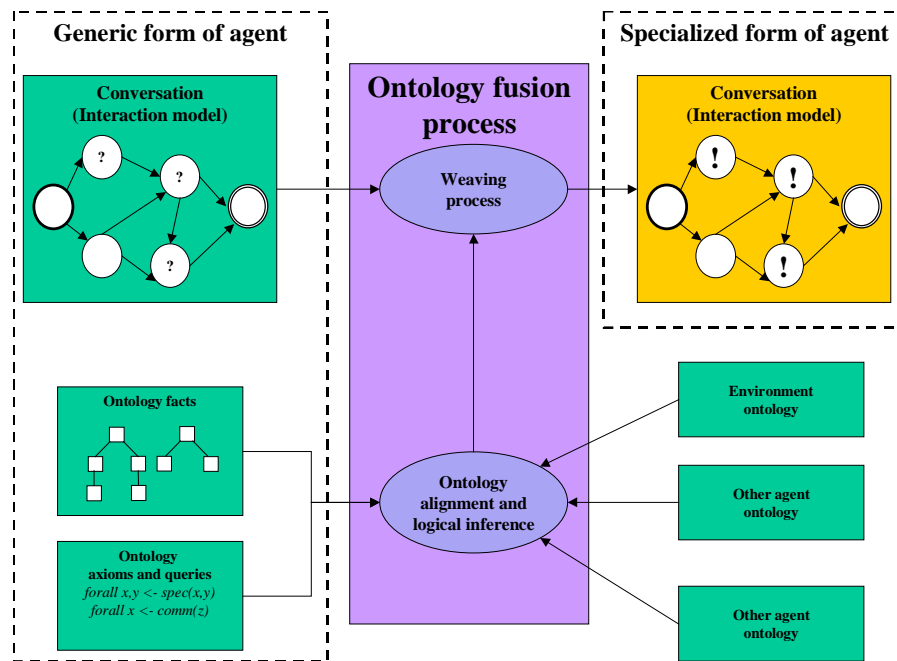


Figure 1. Schema of ontology fusion for agents

Figure 1 shows an agent, which carries some interaction model in the form of a FSM. The agent shall be integrated into some environment by the resolution of certain properties. The properties to be resolved are

indicated with question marks and may describe: method names, server names, resources locations, protocols, etc. Queries to resolve these properties and related axioms and facts are carried by the agent in a form of ontology (Ontology facts and ontology axioms boxes on the above picture). Perform the merge and alignment of the agent ontologies with ontology of surrounding environment and ontologies of one or more other agents. Logical inference process answers given queries resulting into defining all necessary properties. Weaving process substitutes all question marks (undefined information) by just received values and performs the execution of an FSM using reflection features (since many properties were not in place during compilation time)

When agents in our example are communicating using CORBA specification and passing messages by remote call of methods of another agent there are two important problems to be resolved. First of all, not all agent names can be known apriori, thus these are the parameter to determine. Names and parameters of methods to call also can be unknown at the design step of an agent This situation occurs very often during the integration of several agents written by different vendors without any standardized APIs or naming methods. Ontology of an agent includes the information about the service and queries needed to resolve the name of agent which can provide needed service and the name of method with parameters.

Environment ontology contains the exact information about services, which can perform certain actions. Alignment of ontologies allows construct the joint set of axioms and after we can answer consequently and automatically next questions: is the needed service exist? If yes, what's the name of agent exporting service and name of method executing service? By achieving this information, the DII [DII01] of CORBA specification may be used in order to execute a needed method on remote side.

A frequent case of ontology fusion is the fusion of a generic ontology with a more specified one. We have here one generic ontology with free variables (playing a pattern role for data or for operations), and more specified ontology with some values for those variables (playing the role of a specialization of a pattern). In this case, the fusion of the two ontologies is reduced to the unification of these free variables with the concrete values that can be both data and operations providing the specific representation of a pattern in a given context.

Software implementation

In order to implement our conception and to check the feasibility of our approach, we have implemented the prototype for ontology fusion, based on the integration of mobile agent platforms. Prototype is implemented in Java programming language and uses F-Logic [KIF01] for definition of ontologies. The Grasshopper [GRA01] mobile agent platform and Open Cybele agent platform [OPE01], supporting the Activity Centric Programming [ACP01] were used to implement our CIM within the agent paradigm. Ontoprise SiLRI [ONT01] inference engine is used for F-Logic inference. We've modified a standard SiLRI distribution in order to support Java serialization, required for the mobility process.

The system is implemented as a set of Java packages, performing the next set of features. First of all, we provided basic services for some client and some service (which may be agents, CORBA components or something else) to support the carrying, reading and saving of ontologies in the form of RDF for facts and F-Logic text for axioms. Any service contains an F-Logic solver, and may be requested by the client for logical inference. It's necessary to mention, that we don't provide any lookup service, since it exist almost in any environment.

Client side contains client facts and a set of queries, which shall be satisfied. Service side contains the ontology of this service, dedicated to its semantic definition. After the alignment, the binding of variables in queries to specific values is returned by the solver for each query. These variables contain certain information, requested by the client. Interpretation of this information after is up to a client, but we provide some specific patterns for commonly used actions, such as calls of remote method by returning of the method and service name and many others.

Using of our approach for mobile agents paradigm allows using various software patterns. One of them is implemented in a prototype version. This is Conversation Graph Pattern (CGP). The conception of conversation refers to [BAK01] and represents the conversation among agents as a finite-state machine. We provide an implementation of finite state machine executor. The developer may fill states of the FSM by the locations of specific services and pointing to their ontologies and taking into account, that client owns the CGP and has its ontology, the pattern will execute the ontology fusion process among the client and each state service ontology and perform the necessary actions, which are also programmed by the end user. Thus, we provide in the prototype, the proactive model for agents, which support the automated execution, based on the fusion of ontologies. In the example, given above, the implementation takes few seconds using the graph pattern and ontology fusion process.

Comparison with other approaches

From the architectural point of view "Ontology Fusion" approach to integration of entities in a distributed system correlates naturally with such research directions as reflective middleware [COU01] and Aspect-Oriented programming [KIC01]. We use the same concept of reflection as in a reflective middleware dividing "what to do" from "how to do". The weaving metaphor to specialize and modify behavior of a software entity was gained from the Aspects architecture. However comparison with such known representative implementation of reflective middleware as SAMcode [NOR01] and Aspect-Oriented toolkit PROSE [POP01] shows unique attractive features of our approach. Strict programmatic approach to define agent conditional activities and modifiable behavior in SAMcode or cross-cut concepts in PROSE does not provide an easy formal way for integration of different loosely coupled entities by alignment of semantic information from them. Joint using of declarative representation of knowledge in form of ontology and inference logical procedures allows to reduce coding efforts and to provide logical verification of runtime modification of entity behavior.

Ontology-mediated business integration is proposed in [OME01]. This work describes the construction of mediating ontology for the use in XML-based integration environments such as Web Services WDSL or PSL. Integration of services is proposed by the construction of a mediation ontology representing all the objects which can appear during the execution. RDF Schema and prolog is used for ontology description and reasoning. This work doesn't define however any mechanism of service execution or reflection relying on the underlying WDSL protocol. This work proposes also keeping all the ontological information inside mediation service. Ontology Fusion uses only the part of ontology needed for certain operation performing thus is easier to construct and manage.

In our opinion there is another result of this work related with semantics interoperability research. DAML researchers developed the DAML-based Web Service ontology, as well as supporting tools and agent technology to enable automation of services on Semantic Web [ANU01]. DAML-S [DAM01] supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. However DAML-S doesn't define any methodology for complex user-defined query manipulating. Our methodology extends DAML-S concept with alignment and query manipulation framework. We have tested switching from SiLRI to FaCT using DAML+OIL as basic ontology description language.

Conclusions and future plans

It was found very perspective to use logically based ontology manipulation and software patterns together. It allows solving different important problems. For example, to cut it short, the application of a conversation graph pattern and ontology allows coarse-grained representation of strong mobility feature for mobile agents systems. We plan to publish an article dedicated to this topic.

There are many different research directions that should be worked on before the matured common integration methodology will be finally designed. It would be desirable to examine other mechanisms instead of F-Logic for description of ontologies, relations and queries, based on the same formal approach for ontology fusion. Choosing F-Logic formalism we have taken into account the most pragmatic reasons (availability, easy to use). For future research other formalisms can be concerned also (for example DAML+OIL). Some authors propose using more complex algorithms for alignment that use some kind of linguistic heuristics and interaction with a user. In future we will be able to consider such algorithms for our problem as well. Investigation on mediation code generation will be done in the nearest future and an open implementation of the Ontology Fusion framework will be published.

We plan the evaluation of the UML [UML01] or formal UML extensions [BER01] for the description of ontologies and generation of F-Logic axioms. This will simplify the use of ontology fusion and reduce the amount of manual coding on logical languages.

References

[ACP01] Activity Centric Programming. Online: <http://www.opencybele.org/docs/Users.pdf>

[ANU01] Anupriya Ankolekar, Frank Huch, Katia Sycara, "Concurrent Execution Semantics for DAML-S with Subtypes", to appear in The First International Semantic Web Conference (ISWC), June, 2002.

[BAK01] A. Baker, JAFMAS. Java-Based Framework for multi-agent systems, Online: <http://www.ececs.uc.edu/~abaker/JAFMAS/>

[BER01] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo . "Reasoning on UML Class Diagrams using Description Logic Based Systems" In Proc. of the KI'2001 Workshop on Applications of Description Logics. CEUR Electronic Workshop Proceedings

- [CHA01] Hans Chalupsky, "Ontomorph: A Translation System for Symbolic Knowledge", 2000. In proceedings to KR2000 conference.
- [COU01] Geoff Coulson, "What is reflective middleware?", Online: <http://boole.computer.org/dsonline/middleware/RMarticle1.htm>
- [DAM01] DAML-S language. Online: <http://www.daml.org/services/>
- [DAM02] DAML+OIL language. Online: <http://www.daml.org>
- [DEC01] Decker S., Brickley D., Saarela J., Angle J. A query and inference service for RDF. Proceedings of The Query Languages Workshop. W3C, 1998. <http://www.w3c.org/TandS/QL/QL98/>.
- [DII01] CORBA Dynamic Invocation Interface. Online: <http://doc.omg.org/formal/97-10-09>
- [GAM01] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Mass. 1995.
- [GRA01] Grasshopper agent platform. <http://www.grasshopper.de>
- [HOR01] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic SHIQ. In David MacAllester, editor, Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17), number 1831 in Lecture Notes In Artificial Intelligence, pages 482-496. Springer-Verlag, 2000.
- [JEF01] Jeff Z. Pan and Ian Horrocks. Semantic web ontology reasoning in the SHOQ(D_n) description logic. In Proc. of the 2002 Description Logic Workshop (DL 2002), 2002. To appear.
- [KIC01] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin. Aspect-Oriented Programming. In 1997 European Con I. On Object-Oriented Programming (ECOOP '97), pp.220-242. Springer Verlag, 1997.
- [KIF01] M. Kifer and G. Lausen, F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance and Scheme Proc. ACM SIGMOD Intl. Conference on Management of Data , pages 134 - 146, Portland, 1989.
- [MOM01] Message Oriented Middleware, Overview. Online: <http://www.sei.cmu.edu/str/descriptions/momt.html>
- [NOR01] Noriki Amano, Takuo Watanabe. A software model for flexible & safe adaptation of mobile code programs, Proceedings of the workshop on Principles of software evolution, pp. 57-61, Orlando, Florida, 2002.
- [NOY11] Natalya Fridman Noy, Mark A. Musen, SMART: Automated Support for Ontology Merging and Alignment, 1999, Online: <http://citeseer.nj.nec.com/247714.html>
- [OME01] Omelayenko B. "Ontology-Mediated Business Integration". In proceedings of the international conference EKAW 2002.
- [ONT01] SiLRI, <http://www.ontoprise.de>
- [OPE01] Open Cybele agent platform <http://www.opencybele.org>
- [POP01] Andrei Popovici, Thomas Gross, Gustavo Alonso. Dynamic weaving for aspect-oriented programming, Proceedings of the 1st international conference on Aspect-oriented software development, pp. 141-147, Enschede, The Netherlands, 2002.
- [TUR01] Turhin, V.F. On cybernetic epistemology, Systems Research, No 1, pp.3-28, 1993
- [UML01] OMG UML specification. Online: <http://doc.omg.org/formal/01-09-69>
- [WOO01] "Agent-Oriented Software Engineering: The State of the Art", Michael Wooldridge, Paolo Ciancarini, 2000 Online: <http://www.science.unitn.it/~pgiorgio/aose/>