Modeling and realizing interoperability

Manfred A. Jeusfeld , Willem-Jan van den Heuvel, Jeroen Hoppenbrouwers, Kees Leune,

Mike Papazoglou, Hans Weigand, Jian Yang

Infolab, Tilburg University,
P.O.Box 90153, 5000 LE Tilburg, The Netherlands
http://infolab.uvt.nl

Abstract. Interoperability between enterprise applications requires an understanding of the obstacles to interoperability in order to provide methods for overcoming these obstacles. We address the problem from different angles: first, we investigate the benefit of multi-lingual ontologies to overcome language problems between users of an enterprise application. Second, we propose a method to integrate legacy components into new enterprise applications. Third, we research the impact service-oriented computing to enterprise application integration. Finally, we present our results with agent-oriented platforms for integrating autonomous applications.

1. Ontologies for enterprise integration

Matching interfaces by routines for data conversion between applications are not solving the problem of discovering the opportunity for application integration and for addressing semantic mismatches between data and services. One aspect of the semantic mismatch is multi-linguality and heterogeneous data representation. We have developed a scheme for representing data into multi-lingual ontologies that allow to find back information from heterogeneous data sources independently from the original data structures in the sources [1]. The method is based on the notion of attribute concepts like size, color, price etc. These concepts are about properties of objects like products or even services. When a new data source (or a new service) is added to the system, the property description of the new item are linked to the abstract attribute concepts in the ontology. A user (or system) requiring to find the item can use the terms in the ontology to locate the item. Since the ontology concepts have surrogate keys, multiple natural language translations can be attached to them.
The link between data sources to the ontology is enriched by meta data about the data source supplier. This allows tracing back any part of an answer to a location request back to the supplier of the original information. The precision is to the tuple level and loss-less: the original data sources can be reconstructed from the integrated representation and vice versa.
The approach has been successfully employed in the MEMO project [2], which produced a portal for B2B commerce for vertically integrated industries. Specifically, the goal was to integrate product and service catalogs from companies of several countries belonging to the same vertical market. Those companies partially share ontologies on how to describe products and services. However, each role of a company in the vertical supply chain induces a specialized sub-ontology that can differ from country to country due to their cultural and legal heritage. This type of heterogeneity is a great obstacle to application integration. For example, the concept of a fire resistance of a floor cover is has different interpretations in different countries due to their national laws and standards. Our solution addressed this problem by allowing to let multiple ontologies co-exist in the integrating system. The user of the system has associated to her the ontologies that apply to her. Whenever she used the system, only her ontologies are active and she only gets information back that is classified into these ontologies.
The approach has been realized using the ConceptBase system [3]. ConceptBase is a multi-user repository system based on the Telos knowledge representation language [4]. Its key features are

- meta modeling: objects are instances of classes, classes are instances of meta classes, meta classes are instances of meta meta classes etc. The abstraction hierarchy is virtually unlimited allowing to model multiple representation schemes into a single uniform framework

- advanced query language: the query language is based on logical deduction. Save recursion over negation is supported as well as aggregation functions. Answers can be returned in XML or other text-based formats

2. Legacy integration

Web-services are rapidly becoming the de-facto technology for developing and integrating highly distributed business applications that are capable of supporting cross-organizational business processes. Web-services can be defined as self-describing, interoperable and reusable business components that can be published, dynamically combined and invoked through the Internet, even when they reside behind a company's firewall.

Web Service Technology is facing a number of daunting obstacles before it can actually deliver its tantalizing promises of massive reuse and seamless cross-enterprise integration. One of the most severe obstacles is the integration of newly developed constellations of web services with existing enterprise legacy systems. These systems are hard to maintain and adapt to new business requirements, but support the value-adding business operations of many brick-and-mortar companies. Existing web-service based solutions for legacy integration are based on the rather unrealistic assumption that legacy systems can be wrapped and reused integrally without any major modification. In addition, they are predominantly technology focused while largely neglecting business process semantics. Lastly, a method for integrating cohesive fragments of legacy system and data with new web-services is currently lacking. At the INFOLAB we have been developing a comprehensive method, named BALES, that encompasses a wide range of techniques and a supporting prototypical toolset for systematically designing and constructing web-service based, cross-organizational business applications with wrapped legacy systems[13]. The method is business-process driven and supports the notion of selective integration of semantically enriched portions of encapsulated legacy application logic and corresponding data.

3. Service-oriented computing

Service-Oriented Computing (SOC) has been emerging as a new computing paradigm that utilizes services as fundamental elements for developing loosely coupled distributed applications/solutions. The platform neutral nature of SOC paradigm holds the promise for interoperability support based on the way the distributed software applications are developed and deployed in the SOC environment. In order for the realization of the full potential of SOC, many issues need to be addressed, which are best presented and described in the Extended Service Oriented Architecture (ESOA) [5]. The ESOA is a layered architecture that utilizes the basic SOA constructs as its bottom layer, on top of which it layers a composition layer and a service management layer. We have been working on the ESOC architecture that provides separate tiers for composing and coordinating services and for managing services in an open marketplace by employing grid services, and use it to streamline and integrate our research activities. Currently we are mainly focusing on the middle layer of the ESOC in the following two areas:

- *Service composition*: the big challenge of service composition lies in the fact that the generation of the composition must be dynamic in terms of service selection and composition construction; and the result of composition must be configurable, manageable and reusable to meet the ever changing business requirements. To support dynamicity, configurability, manageability, and reusability of service composition, we have been working on the following projects:
    - *Business rule driven service composition*: Service composition needs to be flexible in the way that is defined, scheduled, and constructed. It also needs to adapt to changes so that compositions do not need to be re-generated whenever changes occur. To support various types of service composition, we first need to identify the rules and policies that determine the structure of the composition and prescribe alternative services and provider selection. we then need to separate the rules that govern the composition from the composition specifications. This means developing a rule based mechanism that manages the entire life cycle of service composition. Some initial results can be found in [6, 7]

- *Model driven approach for service composition*: The benefits of adapting system analysis and design methodology in service composition is that we gain much more insights in the process of service composition development so that we can maintain traceability between models and design. In our MDA approach we use UML as the method for designing service compositions. This will enable us to develop technology independent composition definitions, which can subsequently be mapped to specific standard (e.g. BPEL) automatically. Furthermore together with UML we use Object Constraint Language (OCL) to express business rules that can govern and steer the process of service composition. Some results can be found in [8].

- *Service transaction management:* The aim of this project is to develop theory, mechanism, and infrastructure for supporting collaborative transactions in the web service environment. This will lead to a comprehensive framework providing: (1) high-level abstractions for specifying consistency requirements in business context; (2) a mechanism for supporting basic transactional behavior for business transactions, (3) a formal model for consistency and reliability verification, and (4) supporting infrastructures for monitoring, maintaining, and managing consistency In this framework, three levels of transactional properties will be investigated to achieve consistency and reliability: *Transactional capabilities of Web services, Transactional requirements and strategies for Web service coordination,* and *Collaborative business process specific requirements for Web service coordination.* Some initial results can be found in [9].

4. Agent-oriented platforms

Agent-orientation is a new software engineering paradigm. An agent is a software module, just like objects and components are. One major break with the object-oriented tradition is that the modules are considered to be autonomous. This has consequences for the design. The main effort in the design process is how to bring together modules (agents) that were developed independently from each other, typically on the basis of different ontologies, and that will remain independent as well. In other words, the focus is on the design of the coordination [11], rather than on the functionality of the isolated components. The question how this coordination is to be achieved and maintained is precisely the main question in interoperability.

In [10] an agent-oriented design method has been introduced that uses *agreement* and *refinement*. The interoperability question cannot be addressed without considering the infrastructure, or "society" in which the agents meet each other. What should be present, minimally, to make interoperability possible in the first place? What coordination mechanisms should be supported? Reaching agreement (against the background of a certain infrastructure) is another necessary capability for software entities to achieve interoperability. This cannot be done without making explicit what is to be agreed upon. In our research, we have explored this problem under the heading of contract-driven coordination [12]. An expressive formal logic language for contracts is described in [14]. The language is based on deontic logic; this is because dealing with autonomous (rather than completely controllable) systems implies that norms, goals and commitments are not always kept, and the interoperating systems should be able to cope with these violations.

5. References

[1] M.A. Jeusfeld (2004): Integrating product catalogs via multiple-language ontologies. In Proc. EAI 2004 Workshop Enterprise Application, Oldenburg, February 12-13, 2004, online CEUR-WS.org/Vol-93, ISSN 1603-0073.

[2] M.P. Papazoglou, M.A. Jeusfeld, H. Weigand, M. Jarke: Distributed, interoperable workflow support for Electronic Commerce. In *Proc. GI/IFIP Conf. Trends in Electronic Commerce*, Hamburg, Germany, June 3-5, 1998, Springer-Verlag, LNCS 1402, pp. 192-204.

[3] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, M. Staudt, S. Eherer: *ConceptBase - a deductive object base for meta data management. Journal of Intelligent Information Systems* 4(2): 167-192, 1995.

[4] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis: *Telos -- a language for representing knowledge about information systems.* In *ACM Trans. Information Systems*, 8, 4, 1990, pp. 325-362.

[5]: Mike P. Papazoglou and Dimitrios Georgakopoulos (2003): Introduction. In CACM 46 (10): 24-28, Special Issue: Service-oriented computing.
[6] B. Orriens, J. Yang, and M.P. Papazoglou, “A Framework for Business Rule Driven Service Composition”. In VLDB-TES2003, Sep, Berlin, 2003.
[7] Jian Yang and Mike P. Papazoglou, “Service components for managing the life-cycle of service compositions”. Information Systems 29(2): 97-125 (2004).
[8] B. Orriens, J. Yang, and M.P. Papazoglou (2003): Model Driven Service Composition”. In Proc of 1st international conference on service oriented computing (ICSOC2003), Trento, Italy.
[9] Mike P. Papazoglou: “Web Services and Business Transactions”. Journal of World Wide Web 6(1): 49-91 (2003).
[10] Weigand, H., Dignum, V., Meyer, J-J., Dignum, F.: Specification by Refinement and Agreement: Designing Agent Interaction Using Landmarks and Contracts. In: Petta, P., Tolksdorf, R., Zambonelli, F. (Eds.): Engineering Societies in the Agents World III: Proceedings ESAW’02, LNAI 2577, Springer-Verlag, 2003, pp. 257-269.
[11] Zambonelli F., Jennings, N., Wooldridge, M.: Organisational Abstractions for the Analysis and Design of Multi-Agent Systems. In: Ciancarini P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering, LNCS 1957, Springer-Verlag, 2001, pp. 235 – 251.