Infrastructure for Semantic Interoperability and Integration: Breakout Discussion Summary

Mark Burstein and Mike Uschold

Participants:
Naveen Ashish
Mark Burstein
Peter Denno
Yannis Kalfolglou
Mike Uschold

Over the course of the week, one of the questions that came up repeatedly was: "what kind of infrastructure would be needed to support semantic interoperability and integration of systems"? It was widely assumed that semantics would be used for enabling translation and improving interoperability. However, it was not well understood how semantics or semantic mappings were to be *shared*. The emerging notion of a Semantic Web built on the World Wide Web infrastructure has spurred renewed interest in the possibility of semantic interoperability. However, much remains to be done to clearly articulate what the vision of 'semantic interoperability' is exactly, and many fundamental issues need to be addressed before we will see some broad-based theoretical and practical results that will make the vision become a reality. These questions range from the pragmatics of how definitions and mappings are published, shared and found to more fundamental concerns about *what level of description* the shared semantics needs to address.

Architectural Views: conceptual vs. engineering

Our discussion started with a review of a systems engineering approach to integration, which starts by consideration of both the engineering views of the systems to be related (bottom up) and by developing a top-down model of the joint activities that drive the integration effort. It is generally desirable that interoperability occur somewhere between these two different levels of description. Semantic descriptions need to be shared in only as much detail as is required to accomplish the goals of the joint activity. This would exclude many details, say of the internal approaches taken by the two systems or agents. It is at this intermediate level that one seeks to define both the domain objects referenced in discourse between the systems, and also the processes that the systems are coordinating.

Dagstuhl Seminar Proceedings 04391
Semantic Interoperability and Integration
http://drops.dagstuhl.de/opus/volltexte/2005/52

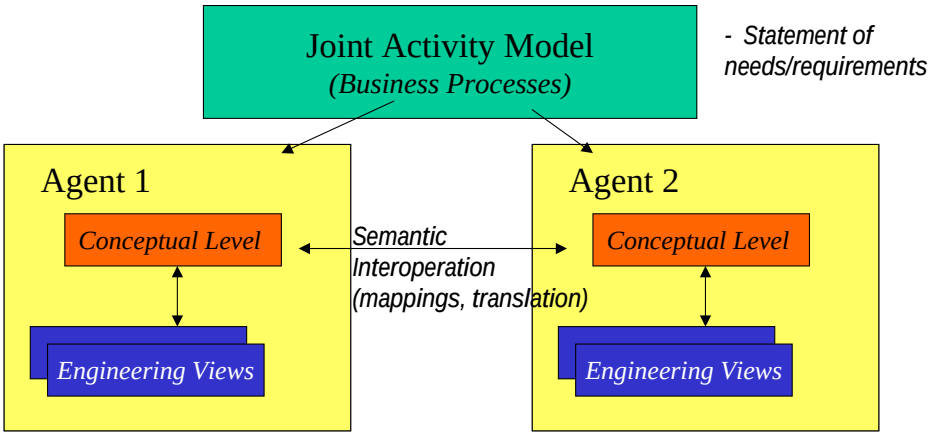

Figure 1. Systems Engineering Approach to Integration

The group agreed that it is important here to distinguish the conceptual level view from the engineering (i.e. implementation) view and its associated data models. The conceptual view must make reference to the purpose of the information, and hide details about data formats that are irrelevant to the interoperation of the systems. For example, one would model a delinquent payment in terms of the activities and actors involved, and not the database filtering view that was used to identify the transaction in a particular database.

Dynamic Interoperability – Self-Describing Agents and Services

A major requirement for semantic interoperability to become more widespread is that it be dynamic; i.e. not relying on the pre-engineering of specific selected systems for interoperation. Instead agents and systems will be *self-describing.* Agents must have the responsibility for characterizing themselves so that other agents can identify ones that they should interact with, and so that they can successfully communicate with them to meet their individual and joint goals. Every agent must, in effect, wear its description 'on its sleeve'.

The translation functions needed to support such interoperability will depend on the level of shared procedures and protocols embodied in the systems, and on the degree to which fundamental ontological concepts (e.g., representations of goals, events, activities and communications) are widely shared. Agents will need to be self-describing at several levels. These levels need to be agreed and shared by the community. The following candidate levels were identified:

- Individual Objectives – The agent-specific objectives that are the objects of inter-agent requests and negotiations.
- Joint Objectives – The goals that drive the joint activities of agents working together. (e.g., contractual agreements and their structure)

- Agent Domain Ontologies – The descriptions of concepts used by agents to specify domain elements referenced in activities, requests and outcomes.
- Agent Protocols – The shared terminology for types of messages exchanged, and the sequencing of those messages to be properly understood.

Architectural Components

These levels give rise to the need for various architectural elements: 1) for handling objectives; 2) for handling ontologies and 3) for handling protocols.

Objectives: Agents must be able to describe their objectives abstractly in ways that enable them to query other agents for potential service partners (service discovery), and to formulate requests to those potential partners, once identified. In addition to agent-specific objectives, it is also necessary to express joint objectives (e.g., contracts, service provision agreements) that are shared by a group of agents working together. Collaborations are formed so agents can better achieve their individual objectives, however the collaborations themselves have working objectives and guidelines that are not specific to any one agent. These joint objectives will be part of a kind of contract that specifies the collaboration goals and contractual agreements (e.g. joint activity models specifying each agent's responsibilities). Both individual and joint objectives need to be *communicated* in an open environment.

As to the architectural elements required to support this vision, we considered two different classes of support within the infrastructure: *community infrastructure* and *protocols*. Examples of community infrastructure agents include service discovery processes facilitated by matchmakers or mediators. Matchmakers act like a 'yellow-pages' query service that simply returns candidates to the requestor. Mediators are services that act as intermediaries, providing a pass-through interface to possibly many back-end services, and addressing the translation and other necessary interoperation issues that that implies.

Other infrastructure agents would include *ontology* services, *mapping lookup* services and *translation* services. Ontology services, which could be combined with mapping services, would support the lookup of concept definitions and other rules or axioms by a query-response protocol. This is needed on the semantic web because the only form of lookup is by URI, while concept descriptions and relationships between concepts can be distributed among many URIs at many different sites. Translation services may come in two distinct types. One kind responds to requests for translations from a single agent (who then forwards the result to its recipient) and the other kind operates as a mediator, sitting between two agents and translating between them.

The other kind of infrastructure support needed facilitate accurate inter-agent communication is support for an extensible variety of protocols. For example, there

will be protocols for interacting with the agents mentioned above (e.g. like service discovery matchmakers), and for negotiating service contracts, and for communicating and processing semantic definitions and mappings themselves.

Sociological Factors

We returned several times to questions about the sociological factors that might help or hinder the evolution of the semantic web and semantic interoperability in open environments. These issues include the issue of privacy, and control of the ontologies, mappings and protocols used in business processes, including any potential competitive advantage of holding these developments closely.

Can we start NOW?

We closed by considering the following hypothetical question:

Suppose you just found out that there was a call for proposals to in the area of infrastructure support for semantic interoperability and integration. There are 24 hours left to submit a two page proposal outline. You have 15 minutes to convince your boss that he should give you the time to write up the proposal. Your boss wants to know: what will you deliver, and how will you demonstrate that it will add value?

This discussion reiterated some of the points above, including the need for protocols of various kinds, tools for defining and extending ontologies and mappings, and infrastructure agents. These are somewhat vague, of course. Can we start now? It depends.

Any effort will be much more successful if it can be grounded in specific use cases and scenarios. Yet these are difficult to identify – as was discovered by the breakout group addressing that topic. To build an actual system, you need to demonstrate it working on something non-trivial. You also need to specify requirements and functionality in much greater detail than is summarized in the above discussion. To date, although much current technology could be adapted and used for this purpose, there seems to be little work to date specifically focused in this area.