Use Case Scenario

Adaptation of Similar Interfaces and Information

Cory Casanave, Data Access Technologies

Rev 0.3 – 2/1/2006

Background

These use case scenarios are intended to define an approach to solving a common problem, adaptation of interfaces and information using ontological methods and a set of capabilities to make use of those methods.

This scenario is based around the common and prevalent problem of adaptation of interfaces between both new and legacy software systems in support of related human processes dependent on those systems.

These scenarios are intended to be part of a set of such scenarios that together define the purpose, scope and requirements for proposed and realized capabilities. These capabilities are expected to include;

- Upper, reference and meta Ontologies
- Model Driven Architecture
- Software designed to support the scenario and make use of these Ontologies to enable interface and information adaptation
- Supporting processes and infrastructure

It should also be noted that a particular problem statement could imply multiple capabilities which could be supported by multiple solution scenarios. Also, a given capability may address more than one problem.

Problem Statement

Information systems are increasingly required to operate in an integrated fashion to support automation of business processes, collaboration between parties, coordinated missions and coordination of information assets. In this environment the concept of an independent application or system is becoming increasingly unrealistic. Solutions are realized with systems of systems and collaboration between systems in support of human endeavors that encompass problems ranging from commerce to network centric warfare.

The state of the art of systems of systems and collaborative solutions is based on the concepts of a lattice of services interacting – also known as a “Services Oriented Architecture”. Various standards, technologies and products have become available to

support a services oriented architecture such as Corba, Message Queuing, EDOC and Web Services.

These solutions work well where there are designed interoperability interfaces and/or standards. In the context of a system, department and sometimes an enterprise or even an industry, sufficient consensus can be reached or fiat asserted to provide for a designed solution. However there are frequently islands of designed solutions, incompatible expression across technologies, and excessive variation points. In addition, legacy systems require adaptation to the designed solution or where there is not a designed solution, directly between legacy systems. Adaptation is time consuming, expensive and error prone.

The set of designed solutions become fragile "in the large" and still require manual adaptation. A primary cause of this fragility is the form of expression used to describe the network of interacting services. These forms of expression share a set of problems;

1. They are typically specific to a particular technology such as XML (the MDA[1] effort of OMG mitigates this concern to some extent).
2. Variation in the semantics of the interfaces. For example, one way or bi-directional interactions, synchronous behavior and data references.
3. The context of interactions is frequently not identified and where it is identified, the purpose of that context is not quantified.
4. The terms used to describe the interfaces are just strings, each in a unique namespace without connection to any other term. These strings are frequently symbols not derived from any vocabulary (E.G. "INVDSLS004") or phrases mangled into unique symbols (E.G. "OrderReadyForProduction")
5. The underlying purpose of an interaction related to a change of state of the interacting parties – such as assets owned, information known, obligations, permissions or activities is frequently unstated. Where it is stated, it is stated in imprecise terms.

Due to the above issues *automated interoperability* of systems *outside of a designed solution* becomes an intractable problem. The intractability of this problem can be demonstrated by attempts by competent engineers to achieve interoperability of independent systems based purely on external specifications. Interactions between experts, negotiations and substantial testing are frequently required to achieve interoperability. If competent engineers can not achieve integration based purely on a

[1] www.omg.org/mda

specification it is doubtful that an automated agent could do better. It should be said that some very well specified systems exist, but it is not the general case.

There are also attempts to formalize information and interfaces to a point of mathematical precision such that there is no question as to the meaning and therefore interoperability is a derived solution. These approaches also share a set of problems;

1. To our knowledge, there are no known production examples
2. There are no generally accepted standards for specification
3. Such specification requires an understanding and quantification of the context of human endeavor that has not yet been achieved.
4. Existing interfaces are not specified to the required level of detail and precision
5. There are very few persons capable of such specification
6. Quantifying an interface to the required degree of detail and precision may be more time consuming and costly than adapting it to a designed solution.
7. It will become difficult for a fully automated solution – particularly a dynamic one, to be trusted to provide a correct solution.

Due to the above problems, integration and interoperability of independently designed solutions is difficult, time consuming, expensive and error prone. This causes inefficiencies in processes, redundancy in capabilities and the associated costs and lost opportunities resultant from those inefficiencies and processes. The resultant proliferation of systems, technologies, specifications, vocabularies, standards, processes and information stores has become a substantial burden to large enterprises and in particular to government agencies – both civilian and defense. This burden is realized in excess costs, failed missions, reduced agility and in the inability to realize desired objectives. In military scenarios interoperability problems have caused loss of life. Failure to correctly specify units caused the loss of a $125 Million mars explorer.

Desired Capability – Integration of Similar Interfaces and Information

The capability desired is to substantially reduce the time, costs and errors associated with the adaptation of designed interfaces and information. This capability should allow for the adaptation of existent interface specifications as well as the specification and implementation of interfaces for new systems. Features of this capability should include;

- The ability to adapt legacy interfaces and information specified in a variety of forms
- The ability to specify new interfaces and information that are more easily integrated

- The ability to capture the semantics of interfaces and information to a point sufficient to substantially reduce the time, costs and errors associated with the adaptation of designed interfaces
- The ability to utilize and integrate human understanding of a problem space
- Human and automated validation of adaptation solutions
- A provision for a growing library of common integration concepts as well as concepts unique to a problem domain
- Practicality for use in high volume, high reliability production environments
- Utilization of existing SOA/Middleware infrastructures
- Reuse, such that specifications and solutions are not "point to point"
- Mechanisms for adaptations to become trusted

The resulting capabilities should allow for systems of systems to provide a collaborative solution to problems based on the ability to quickly and efficiently adapt the interfaces and information of those systems. The initial desired capability will assume that interfaces are similar – that they provide like functionality but that that functionality is specified in incompatible forms and using unique terms. Follow-on capabilities will provide for the adaptation of processes, information and interfaces with more disjoint functionality.

Solution Scenario – Adapting Semantically Grounded Specifications

For any automated and many human interfaces, processes and information stores there exists some form of specification. As stated above, these specifications exists in a variety of languages, levels of detail and precision. The "semantics" of the specifications are represented by arbitrary strings for human interpretation. The basis of the approach is automated assistance in the creation of adapters based on ***semantically grounded*** specifications.

The proposed solution scenario involves a series of capabilities as follows;

- Import specifications into a common infrastructure such that they can be further managed, such as the semantic web stack or a MOF[2].

[2] www.omg.org/mof

- Import instances complying with those specifications into the common infrastructure
- Normalize the semantics used to express the specifications such that concepts of "interface", "type", "Information elements", etc, are represented in the same way – regardless of source. This provides "semantic grounding" of the languages or meta models. This will be done once for each language or formalism and the concepts of that language grounded in a reference ontology of language concepts – this has also been referred to as the "semantic core"[3].
- Create and manage a lattice of reference and upper Ontologies that quantify the concepts found in specifications in the domains of discourse – the terms found in models and interfaces. This set of Ontologies should contain concepts and a set of related terms quantified by the contexts in which such terms and concepts are valid. The set of concepts should be open and extensible as new concepts are discovered over time and accepted by various communities. The creation of the upper Ontologies and core semantic can be provided by ontological experts but the use of the vocabularies should not require unique knowledge.
- Establish relationships between the concepts in the specifications, as identified by arbitrary symbols, to the concepts in the reference and upper Ontologies. Establishing these relationships is the process of "Semantically Grounding" the specification and will typically be a human process with automated assistance. The automated assistance should be a "pluggable" capability such that different techniques can be designed and utilize as the state of the art progresses.
- New specifications should be able to be produced directly from the ontological concepts, thus bypassing the matching of arbitrary symbols.
- Automate the production of a specification for an adapter between the specifications in question. Automated components should be able to match like concepts in specification and produce candidate specifications for adaptation. This specification should then be able to be validated by human participants who may also add or correct axioms or implications. The automated production of an adapter specification should also be a "pluggable" capability such that different techniques can be designed and utilize as the state of the art progresses.
- Automate validation of instances across the adapted boundary as well as the use of instances to aid in the automated processes.

[3] www.semanticcore.org

- Produce the production of adapter implementations using model driven techniques for production infrastructures and the production of test cases based on the instances.

The resultant capability would provide for adaptation of interfaces and information specifications based on the "semantic grounding" of those specifications as suggested above and then the physical adaptation of the systems using ontologically aware software components. This capability could be used to produce components that execute in a "standard" information system environment or could be employed in a much more dynamic environment where adaptations are created dynamically. The same capability would assist in the validation of designed solutions for interoperability against legacy systems.

Advantages of this approach are;

- An open architecture intended to grow
- Ability to integrate multiple forms of information
- Use of "off the shelf" execution technologies
- Value can be achieved with readily available technologies but additional capabilities can be added as they become available. No "new logic" is required.
- Provides the simplicity of a "controlled vocabulary" with the power of Ontologies
- Use of the capability would be practical for persons without an ontological background
- Different specifications grounded in different reference Ontologies could be integrated based on relationships between the reference Ontologies – thus not required a "one true solution". However, arbitrary promulgation of redundant reference Ontologies is counter-productive.

Variation Points

The primary variation points are based on the "semantic distance" between the specifications in question as follows:

- Where specifications are represented in the same language, the requirement for normalization of the language or meta model does not exist, the semantic core would not be required.

- Where the specifications express identical concepts but use different terms the only capability required is matching terms to concepts such as is done in "wordnet®[4]". It should be noted that even limited matching of specifications based on different terms for like concepts would provide substantial leverage.
- Where the specifications express concepts that are subsets of each other only inference across subtype relations is required, a well understood problem. Such capability is found in many modeling paradigms such as UML[5] and most ontological systems, including CL, KIF, RDF[6] and OWL[7].
- Various additional relations between concepts can be "understood" by the ontologically aware components that allow for increasing adaptation in the face of semantic distance. Experience needs to be gained as to what relations can provide for reliable adaptation under various conditions. Optimally the environment will "learn" what is and is not reliable.
- Specifications can not be assumed to be complete or correct as they are produced by humans who are making assumptions not evident in the specification as well as simply making errors. Systems that assume universal correctness or consistency would not function well. Precision, detail, errors and inconsistencies will vary between specifications.

Specific Solution Capabilities Required

To achieve the above capability, certain other capabilities need to be put in place.

- Infrastructure for managing specifications and importing specifications from existing artifacts while retaining traceability
- The structure and "meta ontology" for maintaining a growing set of context specific concepts. Context needs to be related to the context implied in the structure of many specifications.
- The set of meta concepts used for specification – the ontology of architecture, or the "semantic core".

[4] http://wordnet.princeton.edu/

[5] www.omg.org/uml

[6] http://www.w3.org/RDF/

[7] http://www.w3.org/TR/owl-features/

- The set of domain concepts and terms that can be used as the upper and reference Ontologies to ground specifications. These can, perhaps, be imported from existing sources such as wordnet, Sumo[8] or Cyc[9].
- The relations sufficient to ground specification terms in the reference Ontologies need to be specified and the appropriate inference implemented. This should include the capability to deal with phrases that join multiple terms.
- The software design components to assist in the semantic grounding of specifications.
- The software design components to produce adapter specifications.
- The MDA components to generate adapters from these specifications.
- Overall environment and user interface
- Runtime execution support

[8] http://www.ontologyportal.org/

[9] http://www.cyc.com/

Use Case

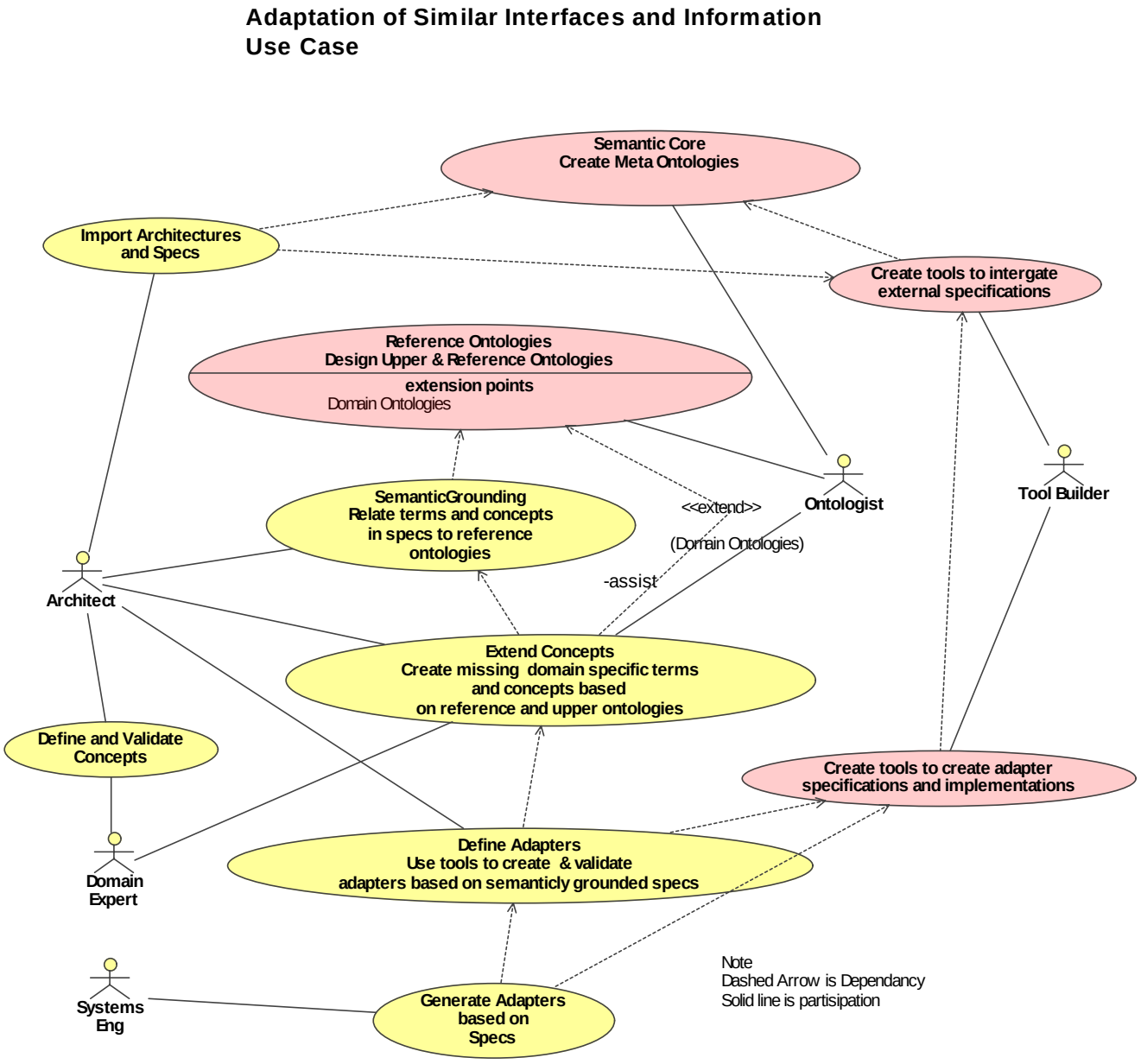

Figure 1

The use case, above, shows the actors and activities relevant to adaptation.

Architect

The architect is using ontologically aware tools to import external models, artifacts and specifications in a common structure based on the "semantic core" ontology/meta model. They then "semantically ground" the specifications by creating relationships to the reference Ontologies. Missing concepts are added, with the assistance of a domain expert, to domain specific Ontologies. Based on this information tools assist in the creation of adapters and information bridges.

The Activity diagram for the architect is as follows:

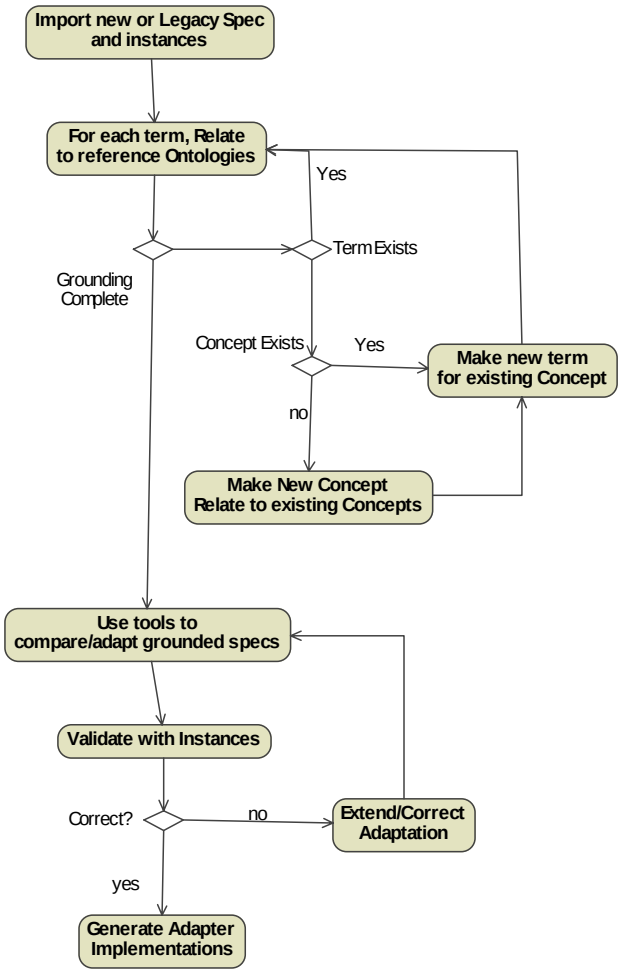

Figure 2

Domain Expert

The domain experts assist in the definition and specification of domain specific terms and concepts.

System Engineer

The systems engineer uses MDA tools to create and deploy adapters.

Ontologist

Ontologists create Ontologies at multiple levels;

- Upper Ontologies define concepts that are useful across a broad range of problems.
- Meta Ontologies define the languages of the Ontologies as well as the concepts in the used to define external specifications.
- Reference Ontologies define common terms and concepts.
- Doman specific Ontologies define concepts specific to a narrow domain of discourse.

Tool Builder

The tool builder provides capabilities to the other actors in the form of software tools, repositories, infrastructure and environments.

Test Cases

A set of test cases is our next step. Our initial test case will be as follows.

Validation and adaptation of finance interfaces

Existing interfaces, implemented by file transfer, have been captured from a legacy financial system. The document structure for these interfaces is represented as Cobol copy books.

A top-down architecture project has produced model-based message structures using UML that are expected to be able to replace the legacy interfaces.

Import both the legacy and UML interfaces and semantically ground them. Provide information as to what elements match up and where there may be discontinuities. For elements that match up, generate specifications for an adapter from the legacy interface to the new interface.

Specific data structures will be provided separately.

