Object Management Group

140 Kendrick Street
Building A Suite 300
Needham, MA 02494
USA

Telephone: +1-781-444-0404
Facsimile: +1-781-444-0320

UML Profile and Metamodel for Services (UPMS)

Request For Proposal

OMG Document: soa/2006-09-09

Letters of Intent due: November 28, 2006
Submissions due: June 4, 2007

Objective of this RFP

Globalization and the Internet have resulted in the need to define more loosely coupled components executing in distributed heterogeneous environments. Service Oriented Architectures (SOA) represent an approach to Information Technology (IT) development that facilitates this loose coupling while at the same time providing sufficient qualities of services necessary for acceptable solutions. There are a number of existing and emerging/evolving runtime platforms resulting in a need to abstract their commonality and define standards for interoperability and information exchange.

This Request for Proposal solicits submissions for a UML Metamodel and Profile for Service (UPMS). Essentially, the UPMS RFP requests a services metamodel and profile for extending UML with capabilities applicable to modeling services using an SOA. The profile will define extensions for modeling and integrating services within and across business enterprises. UPMS will include facilities for formal specification of service contracts that may be developed directly using the profile, or abstracted from business processes. It will also include facilities for indicating which of these contracts are fulfilled by modeled service providers.

Submissions developed in response to this RFP are expected to achieve the following:

- A common vocabulary and metamodel to unify the diverse service definitions that exist in the industry.
- Clarify UML semantics concerned with services modeling and establish modeling best practices.
- Complement existing UML metamodel by defining an extension to UML to ensure complete and consistent service specifications and implementations.
- Integrate with and complement standards developed by other organizations such as W3C and OASIS to facilitate collaborations between service consumer and providers.
- Support a service contract describing the collaboration between participating service consumers and service providers using mechanisms that clearly separate service requirements and specification from realization.
- Enable traceability between contracts specifying services requirements, service specifications that fulfill those requirements and service providers that realize service specifications.
- Facilitate the adoption of Service Oriented Architectures through more abstract and platform independent services models to speed service development, decouple service design from evolving implementation, deployment and runtime technologies, and enable generation of platform specific artifacts.
- The ability to exchange services models between tools using XMI.

It is expected that responses to this RFP will make good use of SOA modeling capabilities already supported by UML. This may require clarifications of UML semantics, best practices that may be incorporated in constraints in UPMS, and possibly issues raised against UML with proposed resolutions.

The purpose of this RFP is to address Service Modeling, not methodologies for SOA. However, submissions are expected to demonstrate how service models relate to business process models on the one hand and existing Web Services standards (XSD, WSDL, BPEL, etc.) on the other in order to facilitate bridging the gap between business models and deployed services solutions.

Submissions developed in response to this RFP should avoid the following capabilities which are expected to be follow-on extensions of UPMS, addressed by separate RFPs, and/or handled by other standards organizations:

- Methodologies for service design. This RFP does not espouse or recommend any service or discovery methodology. Any references to methodologies serve only to enable understanding. What the RFP does specify is the content captured by any service methodology.
- Services governance or compliance.
- Service metrics, policy, security, trust, performance, or other Qualities of Services
- Wire protocols and/or message transfer encodings or marshalling.
- Message delivery reliability, transaction scopes, or other mechanisms for managing data integrity.
- Service brokering, publishing, discovery, service addressing, service registries, asset management.
- Service runtime configuration and deployment.
- Dynamic binding, service federation, mediation, service bus structure, or other service execution concerns.
- User experience or user interfaces.

These are concepts that are not currently addressed in UML or are concerned with qualities of service or service execution. The reason for excluding these capabilities is to limit the scope of this RFP in order to provide a foundation as quickly as possible which can be effectively extended to address these and other concerns as appropriate. Another motivation is to allow these concepts to stabilize in practice before attempting to standardize them, or allow them to be addressed by more appropriate standards bodies.

Adoption of a submission to this RFP will improve communication between modelers, including between business and software modelers, provide flexible selection of tools and execution environments, and promote the development of more specialized tools for the analysis and design of artifacts of emerging Service Oriented Architectures.

For more details, see Section 6 of this document.

1.0 Introduction

1.1 Goals of OMG

The Object Management Group (OMG) is the world's largest software consortium with an international membership of vendors, developers, and end users. Established in 1989, its mission is to help computer users solve enterprise integration problems by supplying open, vendor-neutral portability, interoperability and reusability specifications based on Model Driven Architecture (MDA). MDA defines an approach to IT system specification that separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform, and provides a set of guidelines for structuring specifications expressed as models. OMG has established numerous widely used standards such as OMG IDL[IDL], CORBA[CORBA], Realtime CORBA [CORBA], GIOP/IIOP[CORBA], UML[UML], MOF[MOF], XMI[XMI] and CWM[CWM] to name a few significant ones.

1.2 Organization of this document

The remainder of this document is organized as follows:

Chapter 2 - *Architectural Context* - background information on OMG’s Model Driven Architecture.

Chapter 3 - *Adoption Process* - background information on the OMG specification adoption process.

Chapter 4 - *Instructions for Submitters* - explanation of how to make a submission to this RFP.

Chapter 5 - *General Requirements on Proposals* - requirements and evaluation criteria that apply to all proposals submitted to OMG.

Chapter 6 - *Specific Requirements on Proposals* - problem statement, scope of proposals sought, requirements and optional features, issues to be discussed, evaluation criteria, and timetable that apply specifically to this RFP.

Appendix A – *References and Glossary Specific to this RFP*

Appendix B – General References and Glossary

1.3 Conventions

The key words "**must**", "**must not**", "**required**", "**shall**", "**shall not**", "**should**", "**should not**", "**recommended**", "**may**", and "**optional**" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.4 Contact Information

Questions related to the OMG's technology adoption process may be directed to *omg-process@omg.org*. General questions about this RFP may be sent to *responses@omg.org*.

OMG documents (and information about the OMG in general) can be obtained from the OMG's web site (*http://www.omg.org/*). OMG documents may also be obtained by contacting OMG at *documents@omg.org*. Templates for RFPs (this document) and other standard OMG documents can be found at the OMG *Template Downloads Page* at *http://www.omg.org/technology/template_download.htm*

2.0 Architectural Context

MDA provides a set of guidelines for structuring specifications expressed as models and the mappings between those models. The MDA initiative and the standards that support it allow the same model specifying business system or application functionality and behavior to be realized on multiple platforms. MDA enables different applications to be integrated by explicitly relating their models; this facilitates integration and interoperability and supports system evolution (deployment choices) as platform technologies change. The three primary goals of MDA are portability, interoperability and reusability.

Portability of any subsystem is relative to the subsystems on which it depends. The collection of subsystems that a given subsystem depends upon is often loosely called the *platform*, which supports that subsystem. Portability – and reusability - of such a subsystem is enabled if all the subsystems that it depends upon use standardized interfaces (APIs) and usage patterns.

MDA provides a pattern comprising a portable subsystem that is able to use any one of multiple specific implementations of a platform. This pattern is repeatedly usable in the specification of systems. The five important concepts related to this pattern are:

1. *Model* - A model is a representation of a part of the function, structure and/or behavior of an application or system. A *representation* is said to be *formal* when it is based on a language that has a well-defined form ("syntax"), meaning ("semantics"), and possibly rules of analysis, inference, or proof for

its constructs. The syntax may be graphical or textual. The semantics might be defined, more or less formally, in terms of things observed in the world being described (e.g. message sends and replies, object states and state changes, etc.), or by translating higher-level language constructs into other constructs that have a well-defined meaning. The optional rules of inference define what unstated properties you can deduce from the explicit statements in the model. In MDA, a *representation* that is not *formal* in this sense is not a model. Thus, a diagram with boxes and lines and arrows that is not supported by a definition of the meaning of a box, and the meaning of a line and of an arrow is not a model—it is just an informal diagram.

2. *Platform* – A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

3. *Platform Independent Model (PIM)* – A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

4. *Platform Specific Model (PSM)* – A model of a subsystem that includes information about the specific technology that is used in the realization of that subsystem on a specific platform, and hence possibly contains elements that are specific to the platform.

5. *Mapping* – Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel. A mapping may be expressed as associations, constraints, rules, templates with parameters that must be assigned during the mapping, or other forms yet to be determined.

For example, in case of CORBA the platform is specified by a set of interfaces and usage patterns that constitute the CORBA Core Specification [CORBA]. The CORBA platform is independent of operating systems and programming languages. The OMG Trading Object Service specification [TOS] (consisting of interface specifications in OMG Interface Definition Language (OMG IDL)) can be considered to be a PIM from the viewpoint of CORBA, because it is independent of operating systems and programming languages. When the IDL to C++ Language Mapping specification is applied to the Trading Service PIM, the C++-specific result can be considered to be a PSM for the Trading Service, where the platform is the C++ language and the C++ ORB implementation. Thus the IDL to C++ Language Mapping specification [IDLC++] determines the mapping from the Trading Service PIM to the Trading Service PSM.

Note that the Trading Service model expressed in IDL is a PSM relative to the CORBA platform too. This highlights the fact that platform-independence and platform-specificity are relative concepts.

The UML Profile for EDOC specification [EDOC] is another example of the application of various aspects of MDA. It defines a set of modeling constructs that are independent of middleware platforms such as EJB [EJB], CCM [CCM], MQSeries [MQS], etc. A PIM based on the EDOC profile uses the middleware-independent constructs defined by the profile and thus is middleware-independent. In addition, the specification defines formal metamodels for some specific middleware platforms such as EJB, supplementing the already-existing OMG metamodel of CCM (CORBA Component Model). The specification also defines mappings from the EDOC profile to the middleware metamodels. For example, it defines a mapping from the EDOC profile to EJB. The mapping specifications facilitate the transformation of any EDOC-based PIM into a corresponding PSM for any of the specific platforms for which a mapping is specified.

Continuing with this example, one of the PSMs corresponding to the EDOC PIM could be for the CORBA platform. This PSM then potentially constitutes a PIM, corresponding to which there would be implementation language specific PSMs derived via the CORBA language mappings, thus illustrating recursive use of the Platform-PIM-PSM-Mapping pattern.

Note that the EDOC profile can also be considered to be a platform in its own right. Thus, a model expressed via the profile is a PSM relative to the EDOC platform.

An analogous set of concepts apply to Interoperability Protocols wherein there is a PIM of the payload data and a PIM of the interactions that cause the data to find its way from one place to another. These then are realized in specific ways for specific platforms in the corresponding PSMs.

Analogously, in case of databases there could be a PIM of the data (say using the Relational Data Model), and corresponding PSMs specifying how the data is actually represented on a storage medium based on some particular data storage paradigm etc., and a mapping from the PIM to each PSM.

OMG adopts standard specifications of models that exploit the MDA pattern to facilitate portability, interoperability and reusability, either through ab initio development of standards or by reference to existing standards. Some examples of OMG adopted specifications are:

1. *Languages* – e.g. IDL for interface specification, UML for model specification, OCL for constraint specification, etc.

2. *Mappings* – e.g. Mapping of OMG IDL to specific implementation languages (CORBA PIM to Implementation Language PSMs), UML Profile for EDOC (PIM) to CCM (CORBA PSM) and EJB (Java PSM), CORBA (PSM) to COM (PSM) etc.

3. *Services* – e.g. Naming Service [NS], Transaction Service [OTS], Security Service [SEC], Trading Object Service [TOS] etc.

4. *Platforms* – e.g. CORBA [CORBA].

5. *Protocols* – e.g. GIOP/IIOP [CORBA] (both structure and exchange protocol), [XMI] (structure specification usable as payload on multiple exchange protocols).

6. *Domain Specific Standards* – e.g. Data Acquisition from Industrial Systems (Manufacturing) [DAIS], General Ledger Specification (Finance) [GLS], Air Traffic Control (Transportation) [ATC], Gene Expression (Life Science Research) [GE], Personal Identification Service (Healthcare) [PIDS], etc.

For an introduction to MDA, see [MDAa]. For a discourse on the details of MDA please refer to [MDAc]. To see an example of the application of MDA see [MDAb]. For general information on MDA, see [MDAd].

Object Management Architecture (OMA) is a distributed object computing platform architecture within MDA that is related to ISO's Reference Model of Open Distributed Processing RM-ODP[RM-ODP]. CORBA and any extensions to it are based on OMA. For information on OMA see [OMA].

3.0 Adoption Process

3.1 Introduction

OMG adopts specifications by explicit vote on a technology-by-technology basis. The specifications selected each satisfy the architectural vision of MDA. OMG bases its decisions on both business and technical considerations. Once a specification adoption is finalized by OMG, it is made available for use by both OMG members and non-members alike.

Request for Proposals (RFP) are issued by a *Technology Committee* (TC), typically upon the recommendation of a *Task Force* (TF) and duly endorsed by the *Architecture Board* (AB).

Submissions to RFPs are evaluated by the TF that initiated the RFP. Selected specifications are *recommended* to the parent TC after being *reviewed* for technical merit and consistency with MDA and other adopted specifications and *endorsed* by the AB. The parent TC of the initiating TF then votes to *recommend adoption* to the OMG Board of Directors (BoD). The BoD acts on the recommendation to complete the adoption process.

For more detailed information on the adoption process see the *Policies and Procedures of the OMG Technical Process* [P&P] and the *OMG Hitchhiker's Guide* [Guide]. In case of any inconsistency between this document and the [P&P] in all cases the [P&P] shall prevail.

3.2 Steps in the Adoption Process

A TF, its parent TC, the AB and the Board of Directors participate in a collaborative process, which typically takes the following form:

- *Development and Issuance of RFP*

  RFPs are drafted by one or more OMG members who are interested in the adoption of a standard in some specific area. The draft RFP is presented to an appropriate TF, based on its subject area, for approval and recommendation to issue. The TF and the AB provide guidance to the drafters of the RFP. When the TF and the AB are satisfied that the RFP is appropriate and ready for issuance, the TF recommends issuance to its parent TC, and the AB endorses the recommendation. The TC then acts on the recommendation and issues the RFP.

- *Letter of Intent (LOI)*

  A Letter of Intent (LOI) must be submitted to the OMG signed by an officer of the member organization, which intends to respond to the RFP, confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements. (See section 4.3 for more information.). In order to respond to an RFP the respondent must be a member of the TC that issued the RFP.

- *Voter Registration*

  Interested OMG members, other than Trial, Press and Analyst members may participate in specification selection votes in the TF for an RFP. They may need to register to do so, if so stated in the RFP. Registration ends on a specified date, 6 or more weeks after the announcement of the registration period. The registration closure date is typically around the time of initial submissions. Member organizations that have submitted an LOI are automatically registered to vote.

- *Initial Submissions*

  Initial Submissions are due by a specified deadline. Submitters normally present their proposals at the first meeting of the TF after the deadline. Initial Submissions are expected to be complete enough to provide insight on the technical directions and content of the proposals.

- *Revision Phase*

  During this time submitters have the opportunity to revise their Submissions, if they so choose.

- *Revised Submissions*

  Revised Submissions are due by a specified deadline. Submitters again normally present their proposals at the next meeting of the TF after the deadline. (Note that there may be more than one Revised Submission deadline. The decision to extend this deadline is made by the registered voters for that RFP.)

- *Selection Votes*

  When the registered voters for the RFP believe that they sufficiently understand the relative merits of the Revised Submissions, a selection vote is taken. The result of this selection vote is a recommendation for adoption to the TC. The AB reviews the proposal for MDA compliance and technical merit. An endorsement from the AB moves the voting process into the issuing Technology Committee. An eight-week voting period ensues in which the TC votes to recommend adoption to the OMG Board of Directors (BoD). The final vote, the vote to adopt, is taken by the BoD and is based on technical merit as well as business qualifications. The resulting draft standard is called the *Adopted Specification*.

- *Business Committee Questionnaire*

  The submitting members whose proposal is recommended for adoption need to submit their response to the BoD Business Committee Questionnaire [BCQ] detailing how they plan to make use of and/or make the resulting standard available in products. If no organization commits to make use of the standard, then the BoD will typically not act on the recommendation to adopt the standard. So it is very important to fulfill this requirement.

- *Finalization*

  A Finalization Task Force (FTF) is chartered by the TC that issued the RFP, to prepare an *adopted* submission for publishing as a formal, publicly available specification. Its responsibility includes production of one or more prototype implementations and fixing any problems that are discovered in the process. This ensures that the final available standard is actually implementable and has no show-stopping bugs. Upon completion of its activity the FTF recommends adoption of the resulting draft standard called the *Available Specification*. The FTF must also provide evidence of the existence of one or more prototype implementations. The parent TC acts on the recommendation and recommends adoption to the BoD. OMG Technical Editors produce the *Formal Published Specification* document based on this *Available Specification*.

- *Revision*

  A Revision Task Force (RTF) is normally chartered by a TC, after the FTF completes its work, to manage issues filed against the *Available Specification* by implementers and users. The output of the RTF is a revised specification reflecting minor technical changes.

3.3 Goals of the evaluation

The primary goals of the TF evaluation are to:

- Provide a fair and open process
- Facilitate critical review of the submissions by members of OMG
- Provide feedback to submitters enabling them to address concerns in their revised submissions
- Build consensus on acceptable solutions
- Enable voting members to make an informed selection decision

Submitters are expected to actively contribute to the evaluation process.

4.0 Instructions for Submitters

4.1 OMG Membership

To submit to an RFP issued by the Platform Technology Committee the submitter or submitters must be either Platform or Contributing members on the date of the submission deadline, while for Domain Technology RFPs the submitter or submitters must be either Contributing or Domain members. Submitters sometimes choose to name other organizations that support a submission in some way; however, this has no formal status within the OMG process, and for OMG's purposes confers neither duties nor privileges on the organizations thus named.

4.2 Submission Effort

An RFP submission may require significant effort in terms of document preparation, presentations to the issuing TF, and participation in the TF evaluation process. Several staff months of effort might be necessary. OMG is unable to reimburse submitters for any costs in conjunction with their submissions to this RFP.

4.3 Letter of Intent

A Letter of Intent (LOI) must be submitted to the OMG Business Committee signed by an officer of the submitting organization signifying its intent to respond to the RFP and confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements. These terms, conditions, and requirements are defined in the *Business Committee RFP Attachment* and are reproduced verbatim in section 4.4 below.

The LOI should designate a single contact point within the submitting organization for receipt of all subsequent information regarding this RFP and the submission. The name of this contact will be made available to all OMG members. The LOI is typically due 60 days before the deadline for initial submissions. LOIs must be sent by fax or paper mail to the "RFP Submissions Desk" at the main OMG address shown on the first page of this RFP.

Here is a suggested template for the Letter of Intent:

This letter confirms the intent of <___organization required___> (the organization) to submit a response to the OMG <___RFP name required___> RFP. We will grant OMG and its members the right to copy our response for review purposes as specified in section 4.7 of the RFP. Should our response be adopted by OMG we will comply with the OMG Business Committee terms set out in section 4.4 of the RFP and in document omg/06-03-02.

<____contact name and details required____> will be responsible for liaison with OMG regarding this RFP response.

The signatory below is an officer of the organization and has the approval and authority to make this commitment on behalf of the organization.

<___signature required____>

4.4 Business Committee RFP Attachment

This section contains the text of the Business Committee RFP attachment concerning commercial availability requirements placed on submissions. This attachment is available separately as an OMG document omg/06-03-02.

Commercial considerations in OMG technology adoption

A1 Introduction

OMG wishes to encourage rapid commercial adoption of the specifications it publishes. To this end, there must be neither technical, legal nor commercial obstacles to their implementation. Freedom from the first is largely judged through technical review by the relevant OMG Technology Committees; the second two are the responsibility of the OMG Business Committee. The BC also looks for evidence of a commitment by a submitter to the commercial success of products based on the submission.

A2 Business Committee evaluation criteria

A2.1 Viable to implement across platforms

While it is understood that final candidate OMG submissions often combine technologies before they have all been implemented in one system, the Business Committee nevertheless wishes to see evidence that each major feature has been implemented, preferably more than once, and by separate organisations. Pre-product implementations are acceptable. Since use of OMG specifications should not be dependant on any one platform, cross-platform availability and interoperability of implementations should be also be demonstrated.

A2.2 Commercial availability

In addition to demonstrating the existence of implementations of the specification, the submitter must also show that products based on the specification are commercially available, or will be within 12 months of the date when the specification was recommended for adoption by the appropriate Task Force. Proof of intent to ship product within 12 months might include:

- *A public product announcement with a shipping date within the time limit.*
- *Demonstration of a prototype implementation and accompanying draft user documentation.*

Alternatively, and at the Business Committee's discretion, submissions may be adopted where the submitter is not a commercial software provider, and therefore will not make implementations commercially available. However, in this case the BC will require concrete evidence of two or more independent implementations of the specification being used by end- user organisations as part of their businesses. Regardless of which *requirement is in use, the submitter must inform the OMG of completion of the implementations when commercially available.*

A2.3 Access to Intellectual Property Rights

OMG will not adopt a specification if OMG is aware of any submitter, member or third party which holds a patent, copyright or other intellectual property right (collectively referred to in this policy statement as "IPR") which might be infringed by implementation or recommendation of such specification, unless OMG believes that such IPR owner will grant a license to organisations (whether OMG members or not) on non-discriminatory and commercially reasonable terms which wish to make use of the specification. Accordingly, the submitter must certify that it is not aware of any claim that the specification infringes any IPR of a third party or that it is aware and believes that an appropriate non-discriminatory license is available from that third party. Except for this certification, the submitter will not be required to make any other warranty, and specifications will be offered by OMG for use "as is". If the submitter owns IPR to which an use of a specification based upon its submission would necessarily be subject, it must certify to the Business Committee that it will make a suitable license available to any user on non- discriminatory and commercially reasonable terms, to permit development and commercialisation of an implementation that includes such IPR.

It is the goal of the OMG to make all of its technology available with as few impediments and disincentives to adoption as possible, and therefore OMG strongly encourages the submission of technology as to which royalty-free licenses will be available. However, in all events, the submitter shall also certify that any necessary licence will be made available on commercially reasonable, non-discriminatory terms. The submitter is responsible for disclosing in detail all known restrictions, placed either by the submitter or, if known, others, on technology necessary for any use of the specification.

A2.4 Publication of the specification

Should the submission be adopted, the submitter must grant OMG (and its sublicensees) a world- wide, royalty-free licence to edit, store, duplicate and distribute both the specification and works derived from it (such as revisions and teaching materials). This requirement applies only to the written specification, not to any implementation of it.

A2.5 Continuing support

The submitter must show a commitment to continue supporting the technology underlying the specification after OMG adoption, for instance by showing the BC development plans for future revisions, enhancement or maintenance.

4.5 Responding to RFP items

4.5.1 Complete proposals

A submission must propose full specifications for all of the relevant requirements detailed in Chapter 6 of this RFP. Submissions that do not present complete proposals may be at a disadvantage.

Submitters are highly encouraged to propose solutions to any optional requirements enumerated in Chapter 6.

4.5.2 Additional specifications

Submissions may include additional specifications for items not covered by the RFP that they believe to be necessary and integral to their proposal. Information on these additional items should be clearly distinguished.

Submitters must give a detailed rationale as to why these specifications should also be considered for adoption. However submitters should note that a TF is unlikely to consider additional items that are already on the roadmap of an OMG TF, since this would pre-empt the normal adoption process.

4.5.3 Alternative approaches

Submitters may provide alternative RFP item definitions, categorizations, and groupings so long as the rationale for doing so is clearly stated. Equally, submitters may provide alternative models for how items are provided if there are compelling technological reasons for a different approach.

4.6 Confidential and Proprietary Information

The OMG specification adoption process is an open process. Responses to this RFP become public documents of the OMG and are available to members and non-members alike for perusal. No confidential or proprietary information of any kind will be accepted in a submission to this RFP.

4.7 Copyright Waiver

Every submission document must contain: (i) a waiver of copyright for unlimited duplication by the OMG, and (ii) a limited waiver of copyright that allows each OMG member to make up to fifty (50) copies of the document for review purposes only. See Section 4.9.2 for recommended language.

4.8 Proof of Concept

Submissions must include a “proof of concept” statement, explaining how the submitted specifications have been demonstrated to be technically viable. The technical viability has to do with the state of development and maturity of the technology on which a submission is based. This is not the same as commercial availability. Proof of concept statements can contain any information deemed relevant by the submitter; for example:

“This specification has completed the design phase and is in the process of being prototyped.”

“An implementation of this specification has been in beta-test for 4 months.”

“A named product (with a specified customer base) is a realization of this specification.”

It is incumbent upon submitters to demonstrate to the satisfaction of the TF managing the evaluation process, the technical viability of their proposal. OMG will favor proposals based on technology for which sufficient relevant experience has been gained.

4.9 Format of RFP Submissions

This section presents the structure of a submission in response to an RFP. *All submissions* must contain the elements itemized in section 4.9.2 below before they can be accepted as a valid response for evaluation or a vote can be taken to recommend for adoption.

4.9.1 General

- Submissions that are concise and easy to read will inevitably receive more consideration.
- Submitted documentation should be confined to that directly relevant to the items requested in the RFP. If this is not practical, submitters must make clear what portion of the documentation pertains directly to the RFP and what portion does not.
- The key words "**must**", "**must not**", "**required**", "**shall**", "**shall not**", "**should**", "**should not**", "**recommended**", "**may**", and "**optional**" shall be used in the submissions with the meanings as described in RFC 2119 [RFC2119].

4.9.2 Required Outline

A three-part structure for submissions is required. Parts I is non-normative, providing information relevant to the evaluation of the proposed specification. Part II is normative, representing the proposed specification. Specific sections like Appendices may be explicitly identified as non-normative in Part II. Part III is normative specifying changes that must be made to previously adopted specifications in order to be able to implement the specification proposed in Part II.

PART I

- The name of the RFP that the submission is responding to.
- List of OMG members making the submission (see 4.1) listing exactly which members are making the submission, so that submitters can be matched with LOI responders and their current eligibility can be verified.
- Copyright waiver (see 4.7), in a form acceptable to the OMG.

  One acceptable form is:

  "Each of the entities listed above: (i) grants to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version, and (ii) grants to each member of the OMG a nonexclusive, royalty-free, paid up, worldwide license to make up to fifty (50) copies of this document for internal review purposes only and not for distribution, and (iii) has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used any OMG specification that may be based hereon or having conformed any computer software to such specification."

  If you wish to use some other form you must get it approved by the OMG legal counsel before using it in a submission.

- For each member making the submission, an individual contact point who is authorized by the member to officially state the member's position relative to the submission, including matters related to copyright ownership, etc. (see 4.3)
- Overview or guide to the material in the submission
- Overall design rationale (if appropriate)
- Statement of proof of concept (see 4.8)
- Resolution of RFP requirements and requests

Explain how the proposal satisfies the specific requirements and (if applicable) requests stated in Chapter 6. References to supporting material in Part II should be given.

In addition, if the proposal does not satisfy any of the general requirements stated in Chapter 5, provide a detailed rationale.

- Responses to RFP issues to be discussed

  Discuss each of the "Issues To Be Discussed" identified in Chapter 6.

PART II

The contents of this part should be structured based on the template found in [FORMS] and should contain the following elements as per the instructions in the template document cited above:

- Scope of the proposed specification

- Proposed conformance criteria

  Submissions should propose appropriate conformance criteria for implementations.

- Proposed normative references

  Submissions should provide a list of the normative references that are used by the proposed specification

- Proposed list of terms and definitions

  Submissions should provide a list of terms that are used in the proposed specification with their definitions.

- Proposed list of symbols

  Submissions should provide a list of special symbols that are used in the proposed specification together with their significance

- Proposed specification.

PART III

- Changes or extensions required to adopted OMG specifications

Submissions must include a full specification of any changes or extensions required to existing OMG specifications. This should be in a form that enables "mechanical" section-by-section revision of the existing specification.

4.10 How to Submit

Submitters should send an electronic version of their submission to the *RFP Submissions Desk* (*omg-documents@omg.org*) at OMG Headquarters by 5:00 PM U.S. Eastern Standard Time (22:00 GMT) on the day of the Initial and Revised Submission deadlines. Acceptable formats are Postscript, ASCII, PDF, Adobe FrameMaker, Microsoft Word, and WordPerfect. However, it should be noted that a successful (adopted) submission must be supplied to OMG's technical editors in FrameMaker source format, using the most recent available OMG submission template (see [FORMS]). The AB will not endorse adoption of any submission for which appropriately formatted FrameMaker sources are not submitted to OMG; it may therefore be convenient to prepare all stages of a submission using this template.

Submitters should make sure they receive electronic or voice confirmation of the successful receipt of their submission. Submitters should be prepared to send a single hardcopy version of their submission, if requested by OMG staff, to the attention of the "RFP Submissions Desk" at the main OMG address shown on the first page of this RFP.

5.0 General Requirements on Proposals

5.1 Requirements

5.1.1 Submitters are encouraged to express models using OMG modeling languages such as UML, MOF, CWM and SPEM (subject to any further constraints on the types of the models and modeling technologies specified in Chapter 6 of this RFP). Submissions containing models expressed via OMG modeling languages shall be accompanied by an OMG XMI [XMI] representation of the models (including a machine-readable copy). A best effort should be made to provide an OMG XMI representation even in those cases where models are expressed via non-OMG modeling languages.

5.1.2 Chapter 6 of this RFP specifies whether PIM(s), PSM(s), or both are being solicited. If proposals specify a PIM and corresponding PSM(s), then the rules specifying the mapping(s) between the PIM and PSM(s) shall either be identified by reference to a standard mapping or specified in the proposal. In order to allow possible inconsistencies in a proposal to be resolved later, proposals shall identify whether the mapping technique or the resulting PSM(s) are to be considered normative.

5.1.3 Proposals shall be *precise* and *functionally complete*. All relevant assumptions and context required for implementing the specification shall be provided.

5.1.4 Proposals shall specify *conformance criteria* that clearly state what features all implementations must support and which features (if any) may *optionally* be supported.

5.1.5 Proposals shall *reuse* existing OMG and other standard specifications in preference to defining new models to specify similar functionality.

5.1.6 Proposals shall justify and fully specify any *changes or extensions* required to existing OMG specifications. In general, OMG favors proposals that are *upwards compatible* with existing standards and that minimize changes and extensions to existing specifications.

5.1.7 Proposals shall factor out functionality that could be used in different contexts and specify their models, interfaces, etc. separately. Such *minimalism* fosters re-use and avoids functional duplication.

5.1.8 Proposals shall use or depend on other specifications only where it is actually necessary. While re-use of existing specifications to avoid duplication will be encouraged, proposals should avoid gratuitous use.

5.1.9 Proposals shall be *compatible* with and *usable* with existing specifications from OMG and other standards bodies, as appropriate. Separate specifications offering distinct functionality should be usable together where it makes sense to do so.

5.1.10 Proposals shall preserve maximum *implementation flexibility*. Implementation descriptions should not be included and proposals shall not constrain implementations any more than is necessary to promote interoperability.

5.1.11 Proposals shall allow *independent implementations* that are *substitutable* and *interoperable*. An implementation should be replaceable by an alternative implementation without requiring changes to any client.

5.1.12 Proposals shall be compatible with the architecture for system distribution defined in ISO's Reference Model of Open Distributed Processing [RM-ODP]. Where such compatibility is not achieved, or is not appropriate, the response to the RFP must include reasons why compatibility is not appropriate and an outline of any plans to achieve such compatibility in the future.

5.1.13 In order to demonstrate that the specification proposed in response to this RFP can be made secure in environments requiring security, answers to the following questions shall be provided:

- What, if any, are the security sensitive elements that are introduced by the proposal?
- Which accesses to security-sensitive elements must be subject to security policy control?
- Does the proposed service or facility need to be security aware?

- What default policies (e.g., for authentication, audit, authorization, message protection etc.) should be applied to the security sensitive elements introduced by the proposal? Of what security considerations must the implementers of your proposal be aware?

The OMG has adopted several specifications, which cover different aspects of security and provide useful resources in formulating responses. [CSIV2] [SEC] [RAD].

5.1.14 Proposals shall specify the degree of internationalization support that they provide. The degrees of support are as follows:

a) Uncategorized: Internationalization has not been considered.

b) Specific to <region name>: The proposal supports the customs of the specified region only, and is not guaranteed to support the customs of any other region. Any fault or error caused by requesting the services outside of a context in which the customs of the specified region are being consistently followed is the responsibility of the requester.

c) Specific to <multiple region names>: The proposal supports the customs of the specified regions only, and is not guaranteed to support the customs of any other regions. Any fault or error caused by requesting the services outside of a context in which the customs of at least one of the specified regions are being consistently followed is the responsibility of the requester.

d) Explicitly <u>not</u> specific to <region(s) name>: The proposal does not support the customs of the specified region(s). Any fault or error caused by requesting the services in a context in which the customs of the specified region(s) are being followed is the responsibility of the requester.

5.2 Evaluation criteria

Although the OMG adopts model-based specifications and not implementations of those specifications, the technical viability of implementations will be taken into account during the evaluation process. The following criteria will be used:

5.2.1 Performance

Potential implementation trade-offs for performance will be considered.

5.2.2 Portability

The ease of implementation on a variety of systems and software platforms will be considered.

5.2.3 Securability

The answer to questions in section 5.1.13 shall be taken into consideration to ascertain that an implementation of the proposal is securable in an environment requiring security.

5.2.4 Conformance: Inspectability and Testability

The adequacy of proposed specifications for the purposes of conformance inspection and testing will be considered. Specifications should provide sufficient constraints on interfaces and implementation characteristics to ensure that conformance can be unambiguously assessed through both manual inspection and automated testing.

5.2.5 Standardized Metadata

Where proposals incorporate metadata specifications, usage of OMG standard XMI metadata [XMI] representations must be provided as this allows specifications to be easily interchanged between XMI compliant tools and applications. Since use of XML (including XMI and XML/Value [XML/Value]) is evolving rapidly, the use of industry specific XML vocabularies (which may not be XMI compliant) is acceptable where justified.

6.0 Specific Requirements on Proposals

6.1 Problem Statement

Much has been made of the effects of globalization, rapid change, the Internet and the "Flat World" stimulating business innovation and integration. But this can perhaps be summarized as Business Process Management (BPM) techniques to allow each business unit to focus on their key value while leveraging capabilities provided by others for non-core functions. To do so requires business agility and the ability to integrate business processes in a value chain that crosses department, organizational, and other boundaries.

Achieving this business agility and integration requires business to address many of the same concerns that have driven the evolution of modern information technologies and programming models. These concerns include:

- Complexity Management
- The ability to respond to dynamic change
- Modularity
- Encapsulation
- Separation and integration of concerns
- Deferred commitment
- Solutions through composition of other solutions
- Adaptability
- Reuse

Service Oriented Architectures (SOA) represent the current evolution of computing models from functional decomposition, abstract data types, object-oriented technology and component based development. Each step in this ongoing evolution strives to provide better mechanisms for developing solutions that can be easily adapted for change and reuse. A recurring theme in each evolutionary step is to introduce additional capabilities for separation of concerns, loose coupling, and late binding.

Taken together BPM and SOA represent an opportunity to realize the requirements for business agility and integration in order to participate in optimized value chains. SOA helps align IT systems with business strategy while enabling agile business solutions. To achieve the SOA promise, a company needs to understand how to model processes, services and components and how to tie them all together in a consistent manner.

The interest in BPM and SOA and their promises has already resulted in a proliferation of protocols, specifications, metamodels and tools, many of them incompatible with each other. There are many platforms supporting SOA that are also in flux and evolving rapidly. The OASIS SOA Reference Model has only just been completed. The SOA platform challenge is well exemplified by the chart of WS-* and related specifications at http://www.innoq.com/soa/ws-standards/poster/. As a result of this instability, it will be harder for companies to realize the BPM and SOA potential without costly conversion tools and runtime adapters and mediators.

What is needed is a standard for services modeling that raises the abstraction above the variability in the platforms, enables SOA concepts to be deployed on existing platforms, and provides business integration and interchange to be addressed at the architectural level. This separation of concerns allows business analysts and IT architects to design service solutions with a longer lifespan allowing an organization to choose the best technical platform for implementation. At the same time it prevents existing services solutions from inhibiting platform innovation and evolution that may be necessary to meet business objectives.

SOA is only now reaching sufficient maturity to start developing SOA-specific modeling standards. OMG is uniquely positioned to provide services modeling standards because of the already rich set of specifications from business motivation models to standard runtime platforms tied together through MDA. MDA provides an architectural framework for separating service models from evolving platform implementation technologies.

OMG specifications such as the Business Motivation Metamodel, Organizational Structure Metamodel, Semantics of Business Vocabularies and Rules, and the Business Process Definition Metamodel establish the business motivation and operational requirements that drive the need for services solutions. UML2 supports rich capabilities for component modeling which can be used to model services. But UML2 was not designed to address the unique concerns of an SOA.

Software services extend component based modeling with additional capabilities for addressing the effect of integration across ownership boundaries. This introduces the need to explicitly address contracts that formalize the expected interaction between service consumers and providers without coupling them to particular platforms or implementations that would inhibit business agility. It also results in the need to more directly address distribution, integrity, compensation, security and expected qualities of service. SOA solutions may also be more dynamic with services coming and going as needed and service consumers dynamically binding to the service provider offering the best qualities of service.

The goal of this RFP is to solicit submissions that specify new modeling capabilities that extend UML in order to enable effective service modeling. This has to be done in the context of the broader business and requirements modeling on the one hand, and transformations to runtime platforms on the other. Further information is provided through an example in Appendix C.

6.2 Scope of Proposals Sought

This section describes the scope of proposals sought by this RFP. Specific requirements are set forth in Section 6.5, 6.6 and 6.7, below.

The subsections below are intended to not only specify proposal scope, but to also set a context for understanding the requirements.

6.2.1 Service Oriented Architecture

This RFP calls for a standard for modeling services, i.e., a specification of the modeling elements and their relationships used to define services and their interactions. The intention is that submission could be used for modeling any kind of service—including business and software service.

6.2.2 Services Contracts

Service contracts provide a way of specifying service requirements without constraining the architecture for how those requirements might be realized. Service contracts can be used to sketch key services consumers and providers and their interactions in order to accomplish some stated objective – without having to address unnecessary architectural decisions. A solution architecture consisting of specific service consumers and providers can then be designed after more is known about the requirements that must be met. Service contracts can also realize many use cases providing another flexible means of linking contracts to requirements they fulfill while providing a more formal expression of requirements than is possible with use cases.

Services contracts can represent some abstraction of behavior or requirements of a system. The contract name often indicates the intended behavior. Constraints can represent how to evaluate whether the expected results of the party interactions have been achieved and how they are evaluated. These constraints can be used for example to formalize typical business Key Performance

Indicators (KPIs) necessary to evaluate whether party interactions actually realize the intended business goals and objectives.

A service contract indicates the roles parties or service providers that participate in the contract play, their responsibilities or provided services, the rules for how the parties interact, and the constraints they must meet,

Service providers may fulfill services contracts. Contract fulfillment can be strict or loose. Strict contract fulfillment is formal and verifiable by the model. It means that service providers must conform to the contract. Strict contract fulfillment is useful when service contracts represent formal Service Level Agreements between parties to achieve an objective. Loose contract fulfillment simply indicates the intent of the service provider to fulfill the contract without any formal validation that it has done so. This is especially useful when contracts are informal and provides rich traceability between requirements and their realizations.

Service contracts are optional and may be used to model Service Level Agreements that specify the interactions between service consumers and providers without over-constraining the design of the services that fulfill the contracts.

6.2.3 Service Specification

Where service contracts specify requirements that services solutions must meet, service specifications specify exactly what service consumers and providers do in order to use and/or implement services. Unlike service contracts, service specifications are not architecturally neutral. Instead they are designed based on specific architectural decisions for how services will be provided and what realizations must actually do in order to fulfill services contracts.

The primary purpose of service specifications is to isolate consumers or clients from specific implementations used by service providers. The same service specification may be realized by many different service providers each providing different opportunities for customization, qualities of service, or optional features. These service provider implementations or their deployments may also evolve over time. If clients are directly coupled to these service providers, then every time a provider changes, all clients are directly effected and have to be examined to see if change is required. If service consumers only depend on service specifications, then they are decoupled from such changes in services provider implementations as long as those implementations realize the original specification.

Service specifications are declarations that isolate interacting consumers and providers. The service specification can also contain behaviors that indicate constraints on how implementers must realize provided services. That is, a service specification provides a means for defining complete service specifications that include behavioral rules in addition to static interfaces, operations, preconditions, post conditions, and constraints.

Service specifications are optional and may be used when solutions anticipate variability and change that should be managed through increased decoupling of service consumers and providers. For example, a service provider may change its implementation over time perhaps adding new services. Or different service providers realizing the same service specification may be realized by many service providers having different qualities of service. In both cases, clients can depend on the service specification only isolating them from changes in the underlying service provider implementations.

6.2.4 Service Consumers and Providers

Service providers realize service specifications and/or fulfill service contracts. They may often do this by also being service consumers (or requestors) in that their provided service operations may be implemented as a composition of other service providers.

A service provider may interact with many service consumers and other service providers during its lifetime. Such a service provider will interact with other consumers and providers through specific interaction points that provide increased decoupling for more flexible service configurations and easier change management. For example, a purchase order processor service provider may need to interact with a purchaser, invoice provider, scheduler, and shipper. It would do so through separate interaction points so that a change in interactions with the invoice provider, for example, would be isolated to that interaction point and therefore limit the effect on other consumers and providers.

6.2.5 Service Data

Service consumers and providers often have to exchange data when issuing and replying to service requests. This data is often exchanged across organization boundaries using distributed, secure systems. As such, many of the assumptions about object identity are difficult or impossible to manage automatically. As a result, the responsibility for instance correlation across service requests and response conversations has to be managed explicitly by the participating parties.

Service data is therefore often treated as a data transfer object distinguished from domain data or persistent entities that may be the source and/or target of the exchanged data.

6.2.6 Service Realizations

The same service specification may be realized many different ways over time. Some runtime architectures even allow pluggable implementations that replace existing service providers with new, compatible implementations having additional capabilities or different qualities of service. Service realizations have to be consistent with all of the service specifications they realize and the service contracts they fulfill. UPMS can model complete service implementations, including services that are implemented through the recursive composition and choreography of other services.

Service realizations will often need additional information for addressing interaction points and specifying actual binding information necessary for communication between consumers and providers using particular architectural styles.

6.2.7 Service Organization and Categorization

Services are intended to be reused. Additional information is often needed to organize and categorize services in different ways to address different concerns and for different reuse purposes. Ontologies may also be used to describe service domains.

6.2.8 Customization and Extensibility

Since services are intended to be reused, they must provide some means of managing variability so they can be adapted for specific reuse situations. Service providers often specify variability points that can be used to easily adapt a service for a range of intended usages. There are many ways that variability points can be captured including simple configuration properties, adaptor or mediator patterns, or connections to different required service providers. For more complex situations, generalization and specialization may be used to provide more flexible customization, or to override the behavior of a specific service. In other situations, variability may be captured using template parameters and managed using template (pattern) instantiation.

6.2.9 Compliance

This RFP specifies requirements for minimal, foundation facilities for modeling service specifications and realizations. There can be many different methodologies for how to discover, categorize, develop, maintain, and govern such service specifications and realizations that may be future extensions to submissions meeting the requirements of this RFP. Such extensions should be compliant to all the requirements for this foundation in order to ensure interoperability between service consumers and providers, and tools to produce, integrate, adapt, and execute services.

6.3 Relationship to Existing OMG Specifications

Proposals are expected to be consistent with, extend or, possibly, override the following specifications. In each case, the most recent version is applicable unless the most recent version was adopted less than three months before the final submission to this RFP.

6.3.1 Meta Object Facility 2 ptc/04-10-15

MOF2 is used to specify metamodels, in particular UML2 which is being extended by this RFP. Therefore it is expected that the Service modeling capability metamodel would need the facilities of CMOF rather than EMOF (in the same way CMOF is used by UML2 itself) - though this is not a requirement. This is also indicated through the requirement to reuse portions of the UML2 metamodel which is in CMOF.

6.3.2 UML 2 formal/05-07-05

The Services Profile should, where appropriate (e.g. for component modeling), integrate with or reuse the UML2 metamodel – either 'in the large' or in factored form (using capabilities from InfrastructureLibrary or Superstructure). Profiles contained in submissions should be UML2 Profiles.

6.3.3 Semantics of Business Vocabulary and Rules (SBVR) bei/05-08-01

This covers business concepts and rules expressed in semi-formal natural language with a logic- and MOF-based formal underpinning. This has an even higher (than ODM) overlap with service contracts and specifications. As for ODM, it should be possible to transform and/or trace services models in the services profile to such conceptual models.

6.3.4 MOF Queries Views Transformations ad/05-07-01

It may be used to express the actual transformations from services models to Web Services required in submissions by this RFP. QVT may also be used for tracing levels of abstraction such as Business-Conceptual-Logical-Physical data models (whether automatically generated or not) to candidate service architectures captured using the services profile.

6.3.5 Relationship to ITPMF

Explore the relationship to the IT Portfolio Management Facility (ITPMF) specification (if necessary) to keep it aligned and useful for IT Portfolio Management products

6.3.6 XMI formal/05-09-01

The XMI 2.1 standard specifies the format for metamodel and model interchange.

6.3.7 Reusable Asset Specification (RAS) formal/05-11-02

RAS may be used to describe services models as reusable assets.

6.3.8 UML Profile for Enterprise Distributed Object Computing, ptc/02-02-05

6.3.9 Business Process Definition Metamodel (BPDM) (not yet adopted at time of issuing this RFP) RFP is bei/03-01-06

This covers the definition of business organizational processes. It is expected that these processes could be used to specify requirements that must be fulfilled by solutions using an SOA, and captured using the services model. Service contracts could provide a linkage between architected services solutions and business operational requirements specified in BPDM.

For more information see http://www.omg.org/schedule and http://www.omg.org/technology/documents/spec_catalog.htm.

6.4 Related Activities, Documents and Standards

The following references are provided for reference to other OMG activities and relevant industry specifications. Consistency with these specifications is encouraged.

6.4.1 Ontology Definition Metamodel (ODM) (not yet adopted at time of issuing this RFP) – RFP is ad/03-03-40

This provides coverage of conceptual information modeling in a variety of technologies (RDFS, OWL, Topic Maps, Common Logic) as both metamodels and UML 2 Profiles. It has overlap with service contracts and specifications providing knowledge about a particular domain. It should be possible to transform and/or trace data models in services model to such conceptual models or ontologies.

6.4.2 Organization Structure Metamodel (OSM) (not yet adopted at time of issuing this RFP) – RFP is bei/04-06-05

This metamodel covers organization structures and relationships and contact information. It may therefore be used to describe roles service providers play in a services model and provide a linkage between service providers and the business organizational operations.

6.4.3 Knowledge Discovery Metamodel (KDM) (adopted June 16, 2006) – RFP is lt/03-11-04; submission is ptc/06-06-07.

This metamodel covers data structures from both a persistent (database) and program language perspective. It may be relevant for data exchanged between service consumers and providers, and for persistent data required by service implementations.

6.4.4 OASIS (www.oasys-open.org) has published the Reference Model for Service Oriented Architecture which provides the foundation for the glossary for this RFP. It is based on the Reference Model for Service Oriented Architecture 1.0, Committee Specification 1, issued 2 August 2006.

6.4.5 Information Management Metamodel (IMM) (not yet adopted at time of issuing this RFP) – RFP is ab/05-12-02.

The IMM metamodel is a standard addressing information management needs including:

- MOF2 Metamodel for Information Management (IMM)
- UML2 Profile for Relational Data Modeling, with a mapping to the IMM metamodel and SQL DDL
- UML2 Profile for Logical (Entity Relationship) Data Modeling, with a mapping to the IMM metamodel
- UML2 Profile for XML Data Modeling, with a mapping to the IMM metamodel and XML Schema
- UML2 Profile for Record Modeling, with a mapping to the IMM metamodel and COBOL Copybooks

- A standardized 'Information Engineering' data modeling notation with a mapping to the IMM metamodel

It may therefore be used to describe service data and specific rules for XML exchange of service data in transformations from service models to Web Services.

6.4.6 The following are other non-OMG specifications or documents specific to Web Services that are relevant to this RFP:

- Web Services Architecture, W3C Working Group Note, 11 February 2004. This document defines the Web Services Architecture. It identifies the functional components and defines the relationships among those components to effect the desired properties of the overall architecture.
- XSD 1.0 Specification, See also Web services specifications and EMF which includes an implementation of XSD and an XSD editor. Defines an XML document language for describing data schemas.
- Service Data Objects (SDO) is designed to simplify and unify the way in which applications handle data. Using SDO, application programmers can uniformly access and manipulate data from heterogeneous data sources, including relational databases, XML data sources, Web services and enterprise information systems.
- Web Services Definition Language (WSDL) is a language for describing web services interfaces and bindings to endpoints providing the interfaces.
- Service Component Architecture (SCA) describes a means of defining service components that consist of the wiring together of other service component implementations into deployable assemblies.
- Web Services Business Process Execution Language (WSBPEL) is a leading proposed textual notation for specification of web services processes. See also the eclipse BPEL project.
- Software Services Profile is an IBM developerWorks article describing a UML profile for software services, a profile for UML 2.2 which allows for the modeling of services, service-oriented architecture (SOA), and service-oriented solutions. The profile has been implemented in IBM Rational Software Architect, used successfully in developing models of complex customer scenarios, and used to help educate people about the concerns relevant to developing service-oriented solutions.
- OASIS SOA Reference Model: Organization for the Advancement of Structured Information Standards (OASIS), "Service Oriented Architecture

(SOA) Reference Model," Public Review Draft 1.0, 10 February 2006, http://www.oasis-open.org/committees/download.php/16587/wd-soa-cd1ED.pdf

6.4.7 The following OMG technology processes which are currently underway, address concerns related to this RFP (see more details at http://www.omg.org/techprocess/meetings/schedule/index.html). Submissions should address their relationship to these efforts:

- UML 2.2 Infrastructure and Superstructure

  Submitters should be aware that the design of the UML metamodel may change in UML 2.2, in particular the relation between State Machines and Activity Modeling.

- UML 2.2 Object Constraint Language (OCL)

  Submitters should be aware that a metamodel for OCL 2.1 will be defined in UML 2.2 and that this may relate to conditional expressions for process definitions.

- MOF 2.0 Versioning

  The resulting specification should be compatible with MOF and submitters should be aware of the development of this new version.

- MOF 2.0 Query/View/Transformation

  To the extent the resulting specification relies on or supports alternative views of the metamodel, submitters should be aware of these general capabilities being defined for MOF models.

- MOF Model to Text (not yet adopted at time of issuing this RFP) – RFP is ad/04-04-07

  This can be used to express the actual textual (XSD, WSDL, etc.) transformations required in submissions by this RFP.

- Semantics of Foundational Subset for Executable UML Models RFP, ad/05-04-02.

6.4.8 Other related specifications and activities:

- FEA Service Component Reference Model (SRM) and Services and Components Based Architectures (SCBA)

- ebXML

6.5 Mandatory Requirements

Background material to the following sections can be found in 6.1 and 6.2.

6.5.1 Required Profile

6.5.1.1 Submissions to this RFP shall provide a MOF metamodel and equivalent Profile extending UML for Services Modeling as specified in other mandatory requirements specified in this RFP.

6.5.2 UML Compatibility

6.5.2.1 These metamodel and profile extensions can extend, but not conflict with existing UML semantics for extended metaclasses.

6.5.3 Notation

6.5.3.1 Submissions shall specify icons for stereotype extensions to UML in order to extend the UML notation for services modeling.

6.5.4 Platform Independent

6.5.4.1 Submissions shall express the intent of services models rather than any specific means by which that intent may be realized by some runtime platform.

6.5.4.2 Submissions shall include a non-normative mapping from the Services Profile to Web Services (XSD, WSDL, BPEL, etc.) in order to demonstrate the completeness and applicability of the submission.

6.5.5 Specification of Service Contracts

6.5.5.1 Submissions shall provide a means of specifying contracts (i.e., requirements) that must be met by service specifications or realizations. Service contracts shall include:

a. The ability to specify bi-directional, complex, long-lived and asynchronous interactions between service consumers and providers
b. The functions specified by the contract.
c. The ability to realize UseCases that capture contract requirements.
d. The participants in the contract and the roles they play.

e. The responsibilities of those roles in the context of the contract. It shall be possible to specify these responsibilities using service interfaces, service specifications, or particular service providers (see Section 6.5.13).
f. Connectors indicating possible interactions between roles.
g. Behavioral rules for how the roles interact.
h. Constraints or objectives that must be met.

6.5.5.2 Service contracts shall be consistent with the upcoming BPDM choreography and collaborations.

6.5.6 Service Contract Fulfillment

6.5.6.1 Service specifications and/or providers shall have a means to indicate the service contracts they fulfill and how they provide for the roles in the contract.

6.5.6.2 Submissions shall specify a means for supporting both strict and loose contract fulfillment.

a. Strict contract fulfillment shall mean that a service specification or service provider must be conformant with the service contracts it fulfills. Submissions shall provide a complete definition of conformance which includes, but is not limited to type, behavior, and constraint conformance with the constraints, responsibilities and rules for the role in the contract.
b. Loose contract fulfillment shall mean the modeler warrants the service consumers and providers are capable of playing the indicated roles, but this is not validated by the metamodel, nor do all roles in the contract have to be filled by the service specification or provider.

6.5.6.3 The use of service contracts in any services model shall be optional. Services contracts also need not be fulfilled by any service provider (although tools would be encouraged to report on such contracts).

6.5.7 Service Specification

6.5.7.1 Submissions shall provide a means of specifying service specifications independently of how those services are provided or implemented. Such service specifications shall include:

a. Specification of possibly many Service Interfaces where all operations are considered available in distributed, concurrent systems with no assumptions about global synchronization, control or shared address spaces. (Service interface definition in Glossary B.2)
b. Service Operations in Service Interfaces

c. Operation pre and post conditions, parameters and exceptions
d. Constraints service providers are expected to honor
e. Service interaction points through which service interfaces are provided and required in order to decouple consumers and providers
f. Behaviors as methods of operations provided by the Service Specification indicating the behavioral semantics that must be supported by any realizing service provider.

6.5.7.2 The use of service specifications in any services model shall be optional and are provided for additional decoupling between service consumers and service providers. Developers decide the degree of coupling that is tolerable for their solutions.

6.5.8 Specification of Service Data

6.5.8.1 Submissions shall provide a means of indicating structured service data or information exchanged between service consumers and services providers.

6.5.8.2 Submissions shall provide a means of indicating components of service data representing attachments containing opaque information exchanged between consumers and providers. Such attachments shall provide for an indication of the type of the opaque data such as MIME type where applicable, or may be represented by extensions to UML primitive types.

6.5.8.3 The usage semantics of service data shall make no assumptions with regard to global synchronization, control or shared address spaces.

6.5.9 Synchronous and Asynchronous Service Invocations

6.5.9.1 Submissions shall support synchronous and asynchronous service invocation.

6.5.9.2 Synchronicity shall be a property of the invocation of a service operation, not its specification in a service interface or its implementation in a service provider. That is, a service provider may expect a service to be invoked a certain way, but it is up to clients using the service to determine how the service is to be used. In particular, a service that has output or return parameters may be called synchronously or asynchronously. It is up to the client invoking the service to determine if the output is needed or not.

6.5.10 Service Event Handling and Generation

6.5.10.1 Submissions shall provide a means of designating the ability of a service specification or provider to receive an event.

6.5.10.2 Submissions shall provide a means of generating events targeted at a specific service provider or broadcast to interested providers.

6.5.10.3 Service operations that respond to events shall not have outputs, are always invoked asynchronously, and may raise exceptions.

6.5.11 Service Parameters

6.5.11.1 All parameter types of all operations provided by a service specification or service provider shall be either primitive types or service data in order to minimize coupling between service consumers and service providers.

6.5.12 Service Consumers

6.5.12.1 Submissions shall provide a means of designating a consumer which requires services provided by other service providers.

6.5.13 Service Providers

6.5.13.1 Submissions shall provide a means of designating a service provider.

6.5.13.2 Service providers shall provide and require services only through service interaction points. Service providers shall not directly realize or use service interfaces other than realize service specifications. This ensures isolation and decoupling between consumers and providers.

6.5.13.3 A service provider shall be able to realize zero or more service specifications.

6.5.13.4 A service provider must be conformant to all service specifications it realizes. Submissions shall provide a complete definition of conformance which includes, but is not limited to type, behavior, and constraint conformance. Such conformance rules shall be consistent with the conformance rules for fulfilling service contracts.

6.5.13.5 An interaction point of a service provider shall be able to provide and/or require at least one service interface.

6.5.13.6 A service provider shall be able to specify binding information applicable to all its interaction points for use in connecting to service consumers.

6.5.13.7 A service interaction point shall be able to restrict and/or extend the bindings available through that interaction point overriding the binding information of its containing service provider.

6.5.14 Service Realizations

6.5.14.1 Submissions shall provide a means of specifying the realizations of provided service operations through owned behaviors of the service provider.

6.5.14.2 Submissions shall provide multiple styles for specifying behavior implementations including but not limited to UML activities, interactions, state machines, protocol state machines, and/or opaque behaviors. These may differ from the means used for the same operation in any realized service specification.

6.5.15 Composing Services from other Services

6.5.15.1 Submissions shall specify how services are composed from other services.

6.5.15.2 Submissions shall make no assumptions about or constraints on the number of levels of composed services, or impose arbitrary distinctions between specific composition levels.

6.5.16 Connecting Service Consumers with Service Providers

6.5.16.1 Submissions shall support service channels for connections between usages of service consumers and service providers in the internal structure of some containing element.

6.5.16.2 Submissions shall support different degrees of coupling between connected service consumers and providers by allowing service providers to be specified by any of the following:

a. A service interface (supports minimal coupling by allowing connections to any service providing that service interface)
b. A service specification (supports moderate coupling by allowing connections to any service provider realizing the service specification)
c. A service provider (provides direct coupling to a specific service provider)

6.5.16.3 Submissions shall provide a way for service channels to specify a binding from possible bindings specified on the connected interaction point of the service provider and bindings expected by the interaction point of the connected service consumer.

6.5.17 Customizing and Extending Services

6.5.17.1 Submissions shall specify a means of customizing and extending services that shall include, but not be limited to, the following facilities (as specified by UML):

a. Service specification or provider properties that can be configured by setting values during service development, or deferred to service deployment or runtime.
b. Refinement and redefinition through generalization/specialization.
c. Pattern or template specification and instantiation.

6.5.18 Service Partitions

6.5.18.1 Submissions shall provide a means of organizing service specifications and/or providers into partitions for organization and management purposes.

6.5.18.2 Submissions shall provide a means of specifying constraints on the connections between service partitions.

6.5.19 Service Model Interchange

6.5.19.1 The resulting metamodel and profile shall be made available as an XMI document based on the UML and MOF to XMI mapping specifications and the UML rules for exchanging profiles using XMI.

6.5.19.2 Instances of the service metamodel shall be exchanged using XMI as specified in the MOF to XMI mapping specification. Instances of services models created using the equivalent UML profile shall be exchanged using XMI as specified by the UML rules for exchanging instances of UML models with applied profiles. Submissions shall define interchange compliance levels for each for these XMI document formats.

6.6 Optional Requirements

The following requirements would enhance the value of a specification but are not mandatory.

6.6.1 Proposals may provide additional mappings to existing platforms and languages for service specification and execution.

6.6.2 Submissions may provide a means for specifying the preferred encoding for service data exchange in order to maximize interoperability between service consumers and providers.

6.6.3 Submissions may provide a means for specifying a binding metamodel for capturing structured binding information rather than simple strings.

6.7 Issues to be discussed

These issues will be considered during submission evaluation. They should not be part of the proposed normative specification. (Place them in Part I of the submission.)

6.7.1 Relationship to existing UML metamodel

Proposals shall discuss the relationship of submission model elements and UML2 modeling to demonstrate consistency with the UML metamodel. In particular, the following areas shall be discussed:

- Relationship to UML2 component modeling
- Relationship between service specifications and UML2 «specification» Component
- Clarification concerning options for where services choreographies/communication protocols could be captured:
  - In a delegate part of a component
  - In an owned behavior or a component
  - In a collaboration between parts of another assembly component
  - In the contract of a connector
  - In a port class (the type of a Port)
  - Using nested classes
- Additional constraints needed for effective services modeling
- Formalizing SOA modeling best practices
- Relationship between service contracts and UML2 Collaborations and CollaborationUses as indicators of contract fulfillment
- The relationship between services and UML2 Ports and classes used to type ports.

- The use of UML2 Behaviors (Activity, Interaction, StateMachine, ProtocolStateMachine, and/or OpaqueBehavior) in service specifications and realizations.
- Relationship between service data and possible domain data used in service implementations.

6.7.2 Consistency Checks and Model Validation

Submissions shall discuss how the specification supports automated consistency checks for model validation particularly between service contracts, service specifications, and service realizations.

6.7.3 Applicability to Enterprise Service Bus Runtime Architectures

Submission shall discuss how the specification relates to or supports usage of Enterprise Service Bus (ESB) architectures.

6.7.4 Relationship to UDDI.

Submissions shall discuss how instances of services model may be used to capture information necessary to populate UDDI repositories with information necessary for service discovery and use.

These issues will be considered during submission evaluation. They should not be part of the proposed normative specification. (Place them in Part I of the submission.)

6.8 Evaluation Criteria

6.8.1 Submissions will be evaluated on the degree of formality used to express service models and their semantics and balance between the concerns of understandability, completeness, precision, simplicity, and economy.

6.8.1.1 Submissions that establish UML as the foundation for services modeling and extend (i.e., define a new capability of) UML will be preferred.

6.8.2 Submissions that specify semantics of model element extensions using formal methods such as OCL will be preferred.

6.8.3 Although the mapping from the services profile to Web Services is non-normative, submissions will be evaluated against the completeness of the mapping and the use of OMG MDA principles and related work such as MOF QVT and IMM (for XML Schemas).

6.8.4 Evaluation will be based on the ability to translate from one SOA platform to another using the services profile as an intermediate form.

6.8.5 Clarity of the proposed specification for the purpose of implementing conforming tools for modeling services and ease of reviewing for correctness.

6.8.6 The precision, completeness, compactness, and clarity of the services metamodel.

6.8.7 The ability of services models to be translated to existing runtime platforms in common use.

6.8.8 Clear layering of concepts in the services metamodel required for modeling functional behavior from those that specify optional non-functional properties such as performance or usability.

6.8.9 Consistency with existing UML2 notation. The ability to easily draw notation elements by hand and use them in collaborative modeling sessions.

6.8.10 The ability to support different views of the same model elements with different levels of detail and different layout organizations in order to facilitate comprehension by different stakeholders.

6.9 Other information unique to this RFP

Not applicable.

6.10 RFP Timetable

The timetable for this RFP is given below. Note that the TF or its parent TC may, in certain circumstances, extend deadlines while the RFP is running, or may elect to have more than one Revised Submission step. The latest timetable can always be found at the OMG *Work In Progress* page at http://www.omg.org/schedules/ under the item identified by the name of this RFP. Note that "<month>" and "<approximate month>" is the name of the month spelled out; e.g., January.

Approx Day	**Event or** *Activity*	**Actual Date**
	Issuance recommendation of RFP by TF	*September 26, 2006*
	Approval of RFP by Architecture Board *Review by TC ("Three week rule")*	*September 27, 2006*
0	*TC votes to issue RFP*	*September 28, 2006*
60	*LOI to submit to RFP due*	*November 28, 2006*
120	*Initial submissions due*	*June 4, 2007*
134	*Voter registration closes*	*August 5, 2007*
141	*Initial submission presentations*	*June 27, 2007*
	Preliminary evaluation by TF	*September 4, 2007*
240	*Revised submissions due*	*November 19, 2007*
261	*Revised submission presentations*	*December 12, 2007*
	Final evaluation and selection by TF *Recommendation to AB and TC*	*March 2008*
	Approval by Architecture Board *Review by TC ("Three week rule")*	*March 2008*
330	*TC votes to* recommend specifications	*March 2008*
360	*BOD votes to adopt specifications*	*March 2008*

Appendix A References and Glossary Specific to this RFP

A.1 References

The following documents are referenced in this document: TBC

[BCQ] OMG Board of Directors Business Committee Questionnaire, *http://www.omg.org/cgi-bin/doc?bc/02-02-01*

[EDOC] UML Profile for EDOC Specification, *http://www.omg.org/techprocess/meetings/schedule/UML_Profile_for_EDOC_FTF.html*

[FORMS] Download of OMG templates and forms, *http://www.omg.org/technology/template_download.htm*

[Guide] The OMG Hitchhiker's Guide, Version 6.1, *http://www.omg.org/cgi-bin/doc?omg/2002-03-03*

[MDAa] OMG Architecture Board, "Model Driven Architecture - A Technical Perspective", *http://www.omg.org/mda/papers.htm*

[MDAb] "Developing in OMG's Model Driven Architecture (MDA)," *http://www.omg.org/cgi-bin/doc?omg/2001-12-01*

[MDAc] "MDA Guide" (to be published)

[MDAd] "MDA "The Architecture of Choice for a Changing World™"", *http://www.omg.org/mda*

[MOF] Meta Object Facility Specification, *http://www.omg.org/technology/documents/formal/mof.htm*

[OMA] "Object Management Architecture™", *http://www.omg.org/oma/*

[OTS] Transaction Service, *http://www.omg.org/technology/documents/formal/transaction_service.htm*

[P&P] Policies and Procedures of the OMG Technical Process, *http://www.omg.org/cgi-bin/doc?pp*

[RM-ODP] ISO/IEC 10746

[SEC] CORBA Security Service, *http://www.omg.org/technology/documents/formal/security_service.htm*

[TOS] Trading Object Service, *http://www.omg.org/technology/documents/formal/trading_object_service.htm*

[UML] Unified Modeling Language Specification, *http://www.omg.org/technology/documents/formal/uml.htm*

[UMLC] UML Profile for CORBA, *http://www.omg.org/cgi-bin/doc?ptc/01-01-06*

[UMLM] Chapter 6 of UML Profile for EDOC, *http://www.omg.org/cgi-bin/doc?ptc/02-02-05*

[XMI] XML Metadata Interchange Specification, *http://www.omg.org/technology/documents/formal/xmi.htm*

[XML/Value] XML Value Type Specification, *http://www.omg.org/cgi-bin/doc?ptc/2001-04-04*

These documents (and information about the OMG in general) can be obtained from the OMG's web site (*http://www.omg.org/*). Documents may also be obtained by contacting OMG at *documents@omg.org*. Questions related to the OMG's technology adoption process may be directed to *omg-process@omg.org*. General questions about this RFP may be sent to *responses@omg.org*.

A.2 Glossary

Architecture Board (AB) - The OMG plenary that is responsible for ensuring the technical merit and MDA-compliance of RFPs and their submissions.

Board of Directors (BoD) - The OMG body that is responsible for adopting technology.

Common Object Request Broker Architecture (CORBA) - An OMG distributed computing platform specification that is independent of implementation languages.

Common Warehouse Metamodel (CWM) - An OMG specification for data repository integration.

CORBA Component Model (CCM) - An OMG specification for an implementation language independent distributed component model.

Interface Definition Language (IDL) - An OMG and ISO standard language for specifying interfaces and associated data structures.

Letter of Intent (LOI) - A letter submitted to the OMG BoD's Business Committee signed by an officer of an organization signifying its intent to respond to the RFP and confirming the organization's willingness to comply with OMG's terms and conditions, and commercial availability requirements.

Mapping - Specification of a mechanism for transforming the elements of a model conforming to a particular metamodel into elements of another model that conforms to another (possibly the same) metamodel.

Metadata - Data that represents models. For example, a UML model; a CORBA object model expressed in IDL; and a relational database schema expressed using CWM.

Metamodel - A model of models.

Meta Object Facility (MOF) - An OMG standard, closely related to UML, that enables metadata management and language definition.

Model - A formal specification of the function, structure and/or behavior of an application or system.

Model Driven Architecture (MDA) - An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform.

Platform - A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

Platform Independent Model (PIM) - A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

Platform Specific Model (PSM) - A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform.

Request for Information (RFI) - A general request to industry, academia, and any other interested parties to submit information about a particular technology area to one of the OMG's Technology Committee subgroups.

Request for Proposal (RFP) - A document requesting OMG members to submit proposals to the OMG's Technology Committee. Such proposals must be received by a certain deadline and are evaluated by the issuing task force.

Task Force (TF) - The OMG Technology Committee subgroup responsible for issuing a RFP and evaluating submission(s).

Technology Committee (TC) - The body responsible for recommending technologies for adoption to the BoD. There are two TCs in OMG – *Platform TC* (PTC), that focuses on IT and modeling infrastructure related standards; and *Domain TC* (DTC), that focus on domain specific standards.

Unified Modeling Language (UML) - An OMG standard language for specifying the structure and behavior of systems. The standard defines an abstract syntax and a graphical concrete syntax.

UML Profile - A standardized set of extensions and constraints that tailors UML to particular use.

XML Metadata Interchange (XMI) - An OMG standard that facilitates interchange of models via XML documents.

Appendix B General Reference and Glossary

B.1 References

[OASIS RM] OASIS Reference Model for Service Oriented Architecture 1.0, Committee Specification 1, 2 August 2006.

[EDA] Event Driven Architecture.

[SCBA] Services and Components Based Architectures, A strategic Guide for Implementing Distributed and Reusable Components and Services in the Federal Government.

[XSD] XML Schema Definition language.

[WSDL] Web Services Definition Language

[SCA] Services Component Architecture.

[BPEL] Web Services Business Process Execution Lanague

B.2 Glossary

It is expected that submissions will use terminology as defined in the OASIS Reference Model for Service Oriented Architecture 1.0 [OASIS RM] unless there is some overriding reason not to. This RFP uses similar terms but is not intended to constrain submitters to a particular vocabulary. Rather these terms are intended to express the requirements for the RFP submitters in a unambiguous manner. This section includes some additional terms not covered in [OASIS RM] but used in this RFP.

Service Oriented Architecture (SOA) - an architecture for developing, deploying and managing semantically enriched, loosely coupled, platform-independent work units composed as 'Services' among Providers and Consumers to support business functions.

Service **–** a 'unit of work' performed by a 'Service Provider' to achieve desired capability for a 'Service Consumer' under a 'Contract' optionally facilitated by a 'Service Broker'.

Key Performance Indicator - A specification or constraint for measuring performance to monitory how well a business is achieving its quantifiable objectives. Business systems may allow for trend analysis over time or incorporation of escalation and alerting procedures once a particular threshold or trigger level has been exceeded.

Interaction Point - An endpoint or port through which service consumers connect to service providers through service channels. Interaction points manage dependencies between service consumers and service providers by minimizing the coupling to only the things that have to be known for a particular interaction, separate from other interactions.

Service Interface **–** The service interface is the means for interacting with a service. It includes the specific protocols, commands, and information exchange by which actions are initiated that result in the real world effects as specified through the service functionality portion of the service description. (OASIS)

Service Specification - A specification of what service consumers have to know to use services and what service providers have to do to implement them, separate from any particular implementation. Service specifications define what service operations are provided and required in what service interfaces through what interaction point. They specify behavioral rules or choreography for how service consumers and other service providers must interact for each provided service operation. Service specifications also specify constraints that must be met for the service to meet its objectives, and pre conditions and post conditions for each service operation. Service specifications provide further decoupling between service consumers and particular service providers by allowing service consumers to depend only on the service specification, not any particular implementation. Service providers then realize service specifications with concrete implementations.

Service Level Agreement - Service-level agreements (SLAs) are contracts between service providers and consumers that define the services provided, the metrics associated with these services, acceptable and unacceptable service levels, liabilities on the part of the service provider and the consumer, and actions to be taken in specific circumstances. The structure, precision, and feasibility of service level agreements may vary and establish expectations and impacts on the usage, design, and implementation of service providers that fulfill or participate in the service level agreement.

Service Channel - Represents a connection or communication path between two service interaction points. A service channel establishes a conjugate dependency between the consumer and provider interaction points. This dependency is isolated to the connected interaction points, not the service consumer or provider as a whole. In a conjugate dependency, the consumer interaction point required/provided interfaces must be compatible with the provider interaction point provided/required interfaces in order to connect the interaction points with a service channel. A service channel may have binding information selected from the possible bindings provided by the service provider, and preferred bindings desired by the service consumer. A service channel may also have a direction which indicates which interaction point initiates the conversation protocol between the consumer and provider.

Binding - A Binding specifies message format and protocol details for service operations and service data for a particular interaction point. Such binding information is often needed in service specifications in order to establish exactly how service consumers connect to service providers through service channels in distributed, heterogeneous environments.

Choreography- The sequencing or arrangement of services into a meaningful whole for the purpose of meeting some objective or to provide another service. Control of the choreography is distributed throughout the participants, perhaps based on some prior agreement or formal contract. Contrast with orchestration.

Orchestration – A choreography where the control is owned by a third party mediator or some other agent that manages the interaction between the choreographed participants.

Appendix C Purchase Order Process

This appendix provides an example of a services model to illustrate the capabilities that a submission should be able to support. This example does not illustrate all mandatory or optional requirements and is not intended to suggest any particular submission requirements. The example is captured using IBM Rational Software Architect (RSA). The stereotypes used are only intended to clarify the model and highlight concepts described in this RFP. They are not intended to suggest any particular submission requirement or recommendation.

C.1 Introduction

This example is based on the Purchase Order Process example taken from the BPEL 1.1 specification. The BPEL 1.1 spec does not complete this example: correlation sets aren't defined, the business data is incomplete, and there is no fault handling or compensation. This version of the example includes some made-up data for completeness, in particular data needed for correlation.

The example starts by describing the business motivation which establishes the business goals and objectives that are to be achieved. This is followed by a high-level business process captured using BPMN that expresses the business organizational and operational requirements necessary to meet the goals and objectives. These motivations and process models are not the subject of the RFP, but are provided to establish a context connecting the services to business requirements.

Now that the business requirements are understood, the example proceeds to the development of a services solution that meets these requirements. First the business process is viewed as a service contract. Then service interfaces and service providers are designed and connected together in a manner that meets the business requirements while addressing software architecture concerns.

C.2 Business Motivation

A consortium of companies has decided to collaborate to produce a reusable service for processing purchase orders. The goals of this project are to:

- Establish a common means of processing purchase orders.
- Ensure orders are processed in a timely manner, and deliver the required goods.
- Help minimize stock on hand.
- Minimize production and shipping costs

C.3 Business Organizational Processes

The business analysts from the member companies analyzed the requirements and determined that the following business process (expressed in BPMN) meets the business objectives as well as business operational constraints. Two versions of the business process are given to indicate the range of possible input a software architect might get from a business analyst. The first example represents a simple sketch of the business process that does not contain sufficient detail to create a services contract that could use strict fulfillment. The second example represents a refinement of the first processes that is intended to model a more formal service level agreement between the parties. This model could be used to create a strict services contract.

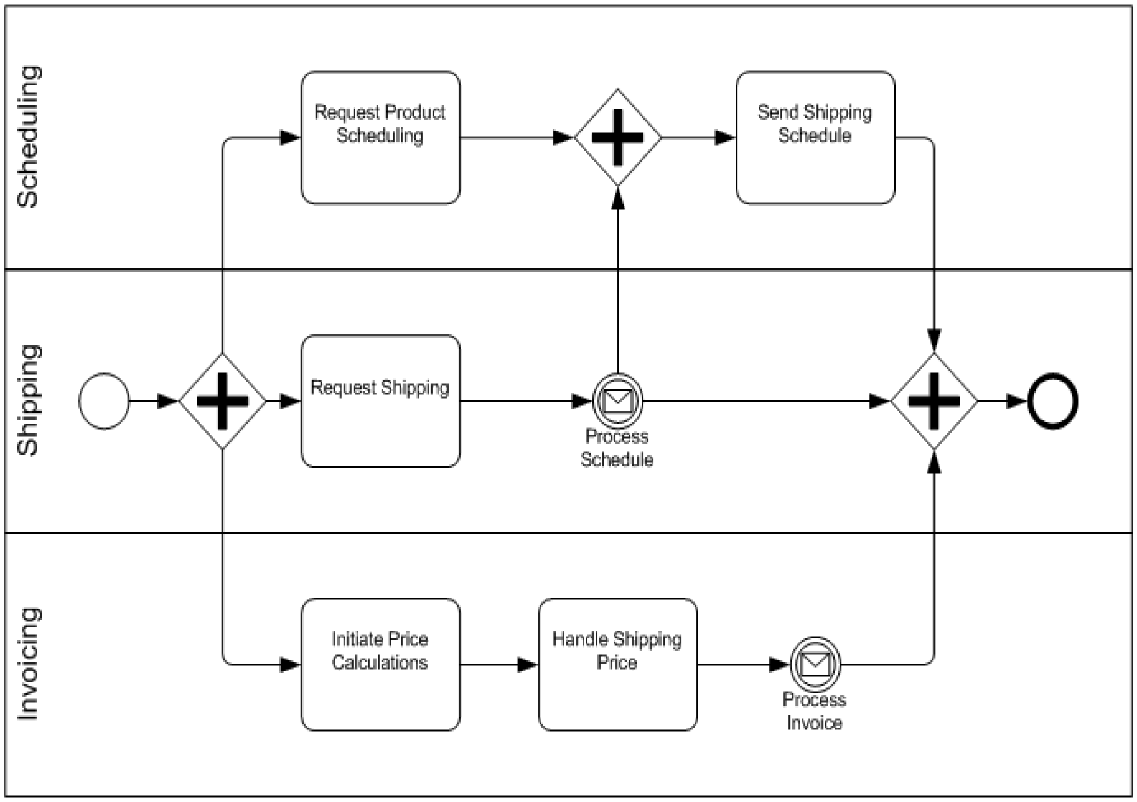


Figure 1: Informal Business Process Model

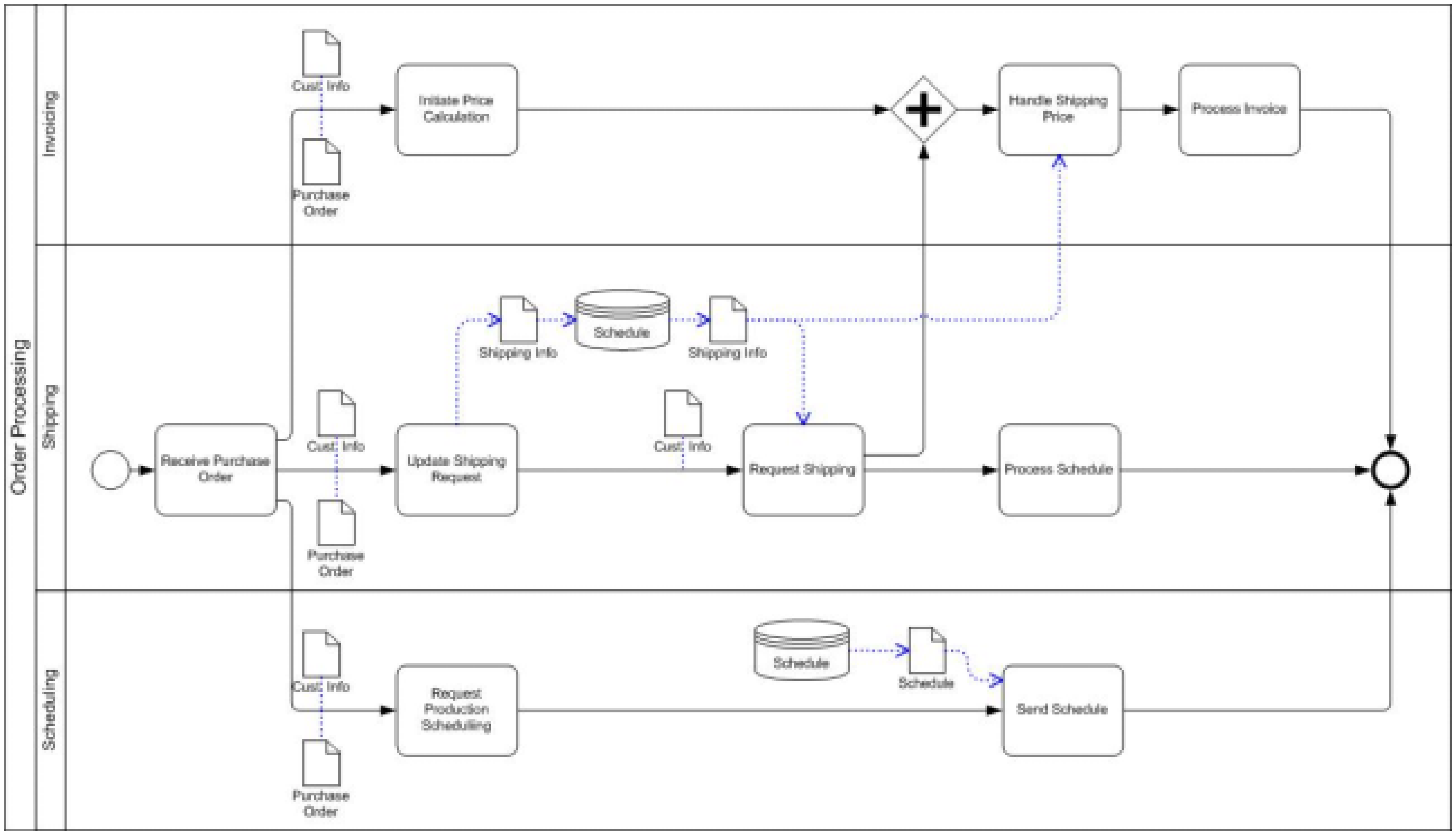


Figure 2: Formal Business Process Model

The purchase order process initiates three parallel activities, one for managing production and shipping scheduling, another for price calculation and invoicing, and a third for shipping the ordered products. Processing starts by initiating a price calculation based on the ordered products. This price is not yet complete however since the total invoice depends on where the products are produced and the amount of shipping costs. At the same time, the order is sent to production in order to determine when the products will be available and from which locations. In parallel, the process requests shipping and then waits for a shipping schedule to be sent from some production scheduling provider. Once the production schedule is available, the invoice can be completed and is returned to the customer.

The next section treats the business process as a requirements contract and shows how to realize an SOA solution for the intended business operational requirements.

C.4 The Services Solution Model

C.4.1 Business Requirements

The requirements for the Purchase Order Process can be obtained from the BPMN business process. These requirements are viewed as a service contract indicating the roles that participate in the business process, the responsibilities they are expected to perform, and the rules for how they interact. This service contract is a formal, architecturally neutral specification of the business requirements that are performed by some interacting service consumers and providers, without addressing any IT architecture or implementation concerns. In this case, the service contract contains the same information as the BPMN process.

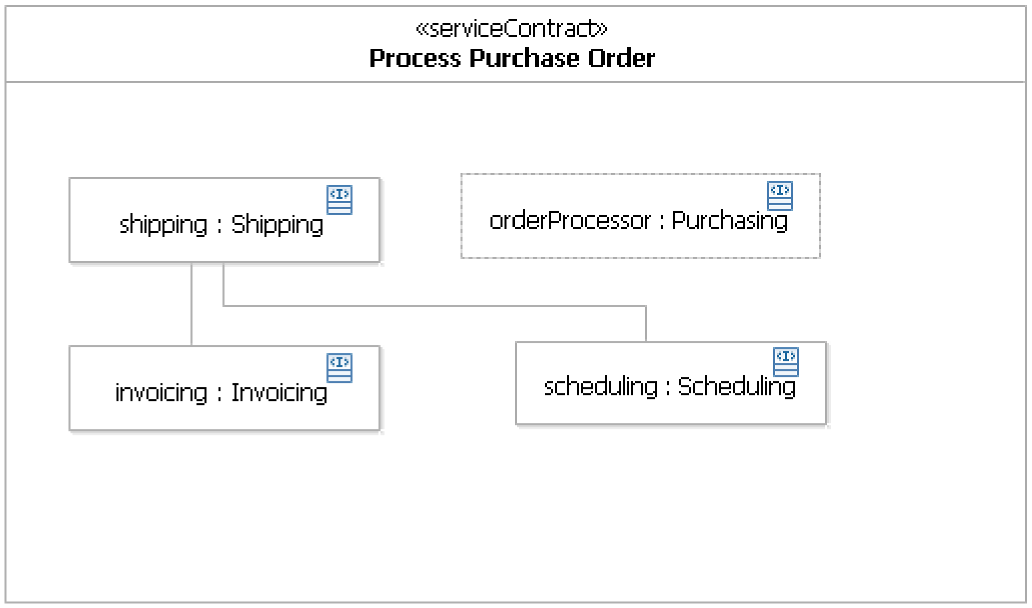


Figure 3: Service Contract – Structural View of Roles and Responsibilities

The following Activity is an ownedBehavior of the Process Purchase Order service contract indicating the rules for how the participating roles interact. The role types indicate the responsibilities they must fulfill as derived from the tasks assigned to the roles in the business process swimlanes.

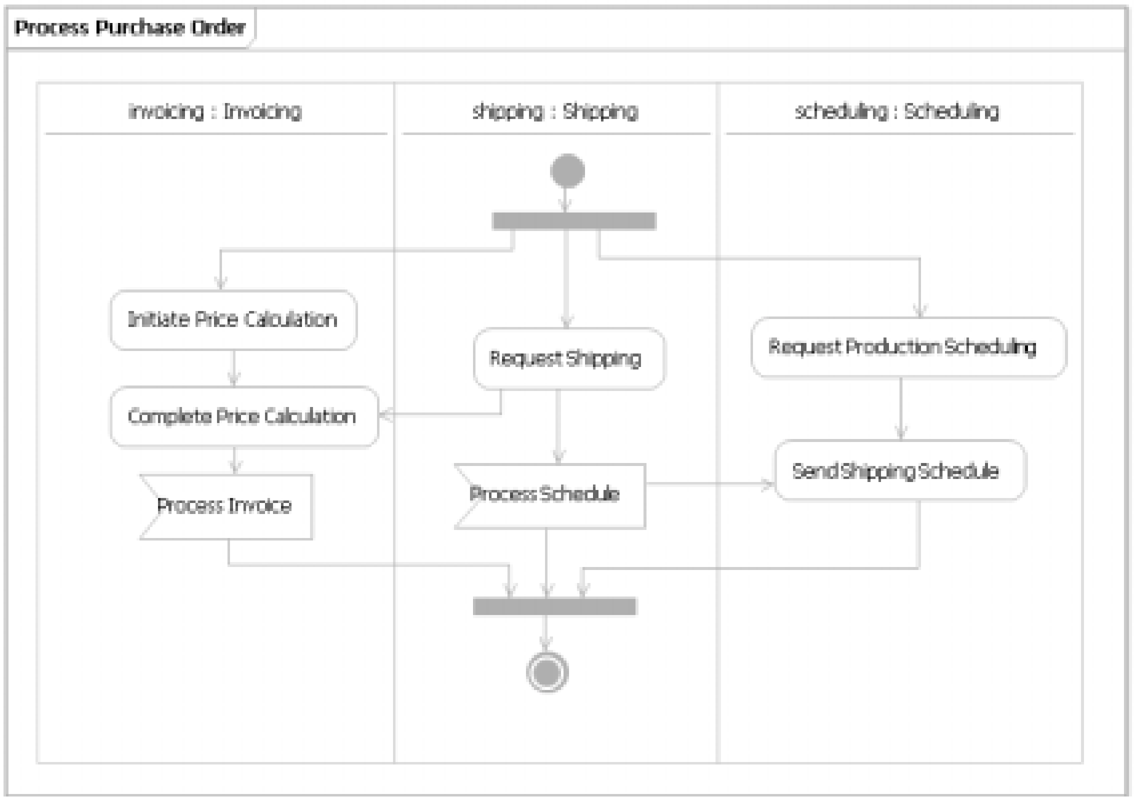


Figure 4: Service Contract – Behavioral Rules View

The Purchase Order Process is a service contract with four roles (one played by the process itself). These roles are derived from the business process by examining the tasks assigned to the roles in the business process swimlanes. BPMN takes a process centered view of business requirements while the services contract takes a role centered view. This provides a more services oriented view of business contracts making it easier to bridge the gap between the process requirements and architected service-oriented solutions.

The service contract may have constraints derived from the metrics and measures from the business motivation model. These constraints indicate what the service contract is intending to accomplish, and how to measure success.

The service contract may also realize use cases that capture additional information about the high-level functional and nonfunctional requirements for the business process from the perspective of the key external stakeholders or actors.

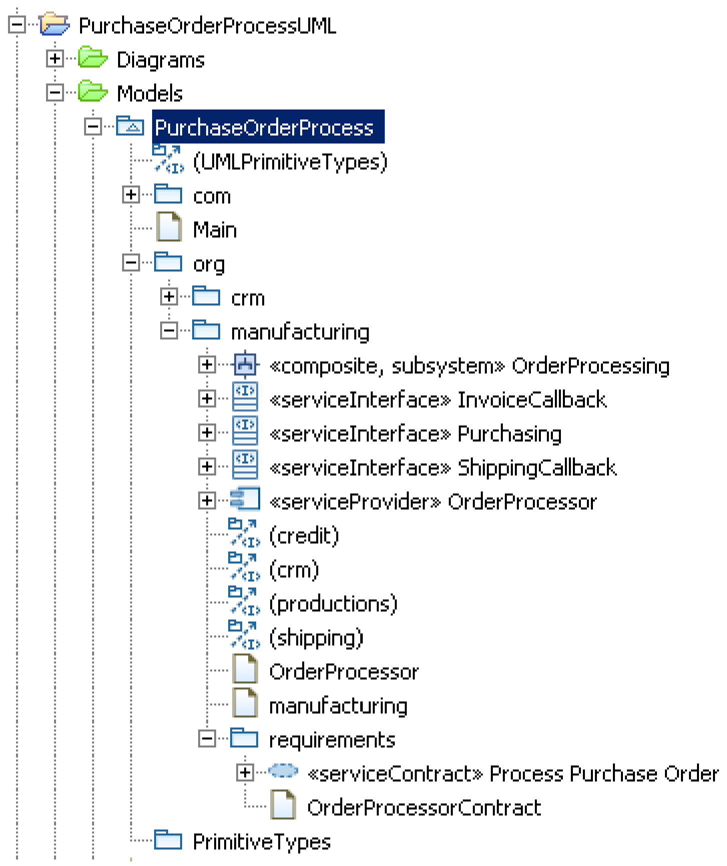
C.4.2 Project Organization

The services model of the PurchaseOrderProcess consists of an information model for the process data, specifications of the provided and required interfaces, and a component providing the order processing services. An architect has decided that the purchasing service will be provided by a component providing a single service operation. This component will fulfill the business operational requirements specified by the services contract.

The example project is organized in a single project called PurchaseOrderProcessUML. The project contains a single model, PurchaseOrderProcess. There are five main packages in the model:

1. **org::manufacturing** – contains the OrderProcessor component which provides the purchase order processing services
2. **org::crm** – contains the data model and common interfaces for some envisioned Customer Relationship Management standard that service consumers and service providers have agreed upon for developing shared services. This and the following packages would probably use a different model for each of the service specifications. In this example they are kept in the PurchaseOrderProcess model in order to keep the example simple.
3. **com::acme::credit** – service interfaces and service providers for credit management services.
4. **com::acme::productions** – services interfaces and service providers for production management and scheduling services
5. **com::acme::shipping** – service interfaces and service providers for shipping services

The overall layout of the PurchaseOrderProcess service model is given in the figure below.



```
PurchaseOrderProcessUML
    Diagrams
    Models
        PurchaseOrderProcess
            (UMLPrimitiveTypes)
            com
            Main
            org
                crm
                manufacturing
                    «composite, subsystem» OrderProcessing
                    «serviceInterface» InvoiceCallback
                    «serviceInterface» Purchasing
                    «serviceInterface» ShippingCallback
                    «serviceProvider» OrderProcessor
                    (credit)
                    (crm)
                    (productions)
                    (shipping)
                    OrderProcessor
                    manufacturing
                    requirements
                        «serviceContract» Process Purchase Order
                        OrderProcessorContract
            PrimitiveTypes
```

Figure 5: The PurchaseOrderProcess Model Organization

The org::manufacturing package defines the OrderProcessor component providing the order processing service. It imports all of the other packages in order to access the CRM standard data and interfaces, and all of its required services.

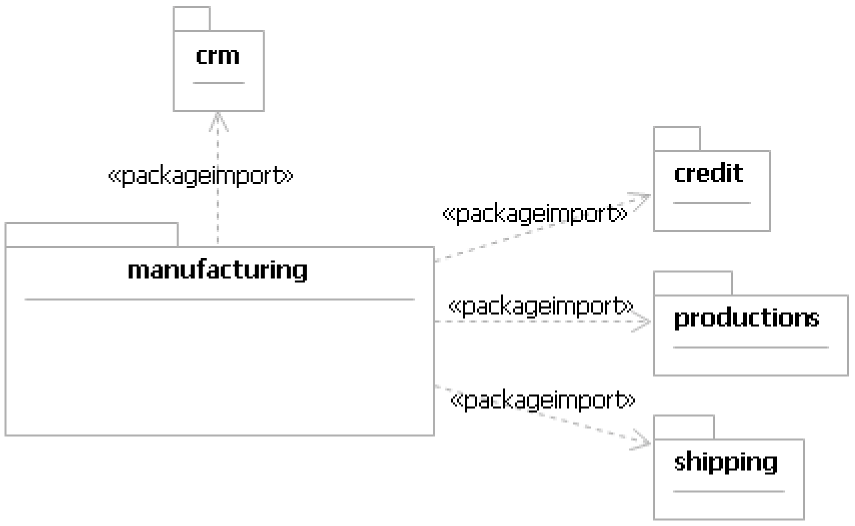


Figure 6: Package Dependencies

C.4.3 Data Model

The Customer Relationship Management data model defined in package org::crm defines all the data used by all service operations in the PurchaseOrderProcess model in this example.

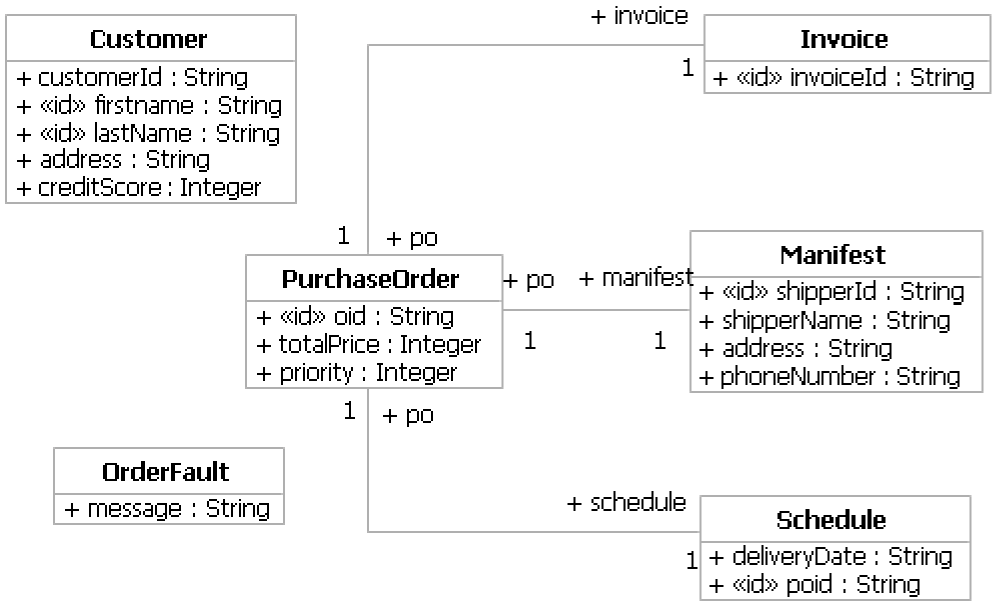


Figure 7: The Services Domain Model

The data model is intentionally kept simple in the example in order to focus on service modeling.

The data model makes use an «id» stereotype to identify attributes that uniquely identify instances of the containing class. This will be a recurring theme throughout the services model since Web Services and BPEL in particular rely on business data for instance correlation.

C.4.4 Required Services

The OrderProcessor requires a number of interfaces from other service providers in order to process a purchase order from a customer. The required services are for invoicing, production scheduling, and shipping. The details of these service providers are not shown to simplify the examples.

C.4.4.1 Invoicing

An invoicing service provider provides the Invoicing interface for calculating the initial price for a purchase order, and then refining this price once the shipping information is known. The total price of the order depends on where the products are produced and from where they are shipped. The initial price calculation may be used to verify the customer has sufficient credit or still wants to purchase the products. It is better to verify this before fulfilling the order.

An Invoicer component provides the Invoicing interface. The invoicing port is a «service» port and can therefore specify the possible bindings provided by the Invoicer component for use in connecting with other components. The invoicing port also requires the InvoiceCallback in order to return information back to the client.

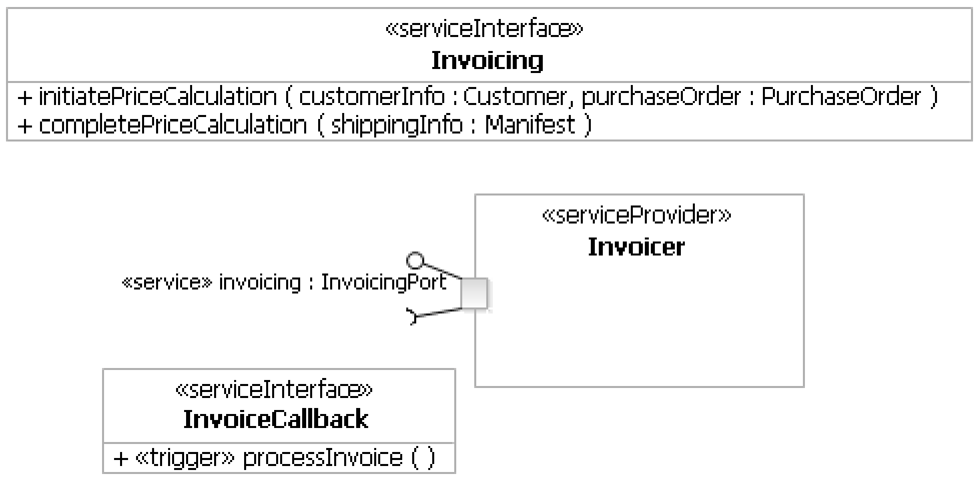

«serviceInterface»
Invoicing
+ initiatePriceCalculation (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ completePriceCalculation (shippingInfo : Manifest)
«serviceProvider»
Invoicer
«service» invoicing : InvoicingPort
«serviceInterface»
InvoiceCallback
+ «trigger» processInvoice ( )

Figure 8: The Invoicer Service Provider

C.4.4.2 Production Scheduling

A production scheduling service provider provides the Scheduling interface to determine where goods will be produced and when. This information can be used to create a shipping schedule used in processing purchase orders.

The Productions component provides the Scheduling interface through its scheduling port. This «service» port specifies the possible binding styles available on that port.

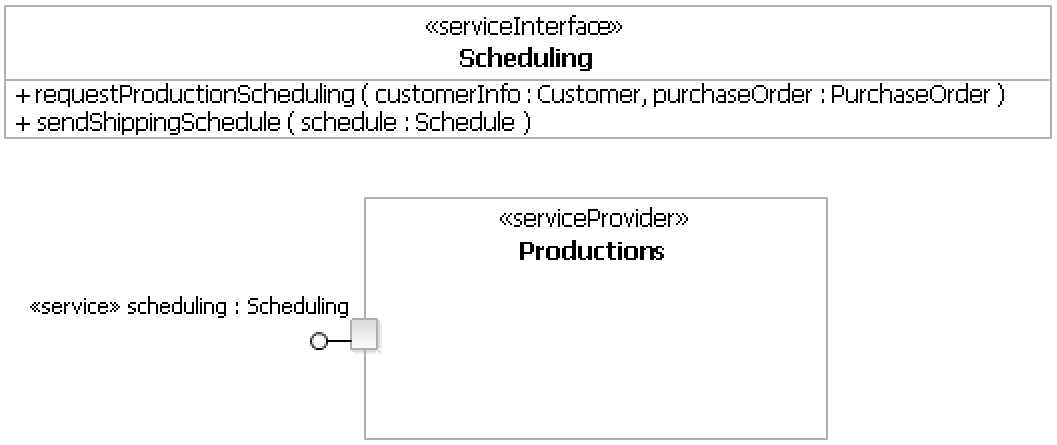

«serviceInterface»
Scheduling
+ requestProductionScheduling (customerInfo : Customer, purchaseOrder : PurchaseOrder)
+ sendShippingSchedule (schedule : Schedule)
«serviceProvider»
Productions
«service» scheduling : Scheduling

Figure 9: The Productions Service Provider

C.4.4.3 Shipping

A shipping service provides the Shipping interface in order to ship goods to a customer for a filled order. It also requires the ShippingCallback in order to send the shipping schedule back to the client.

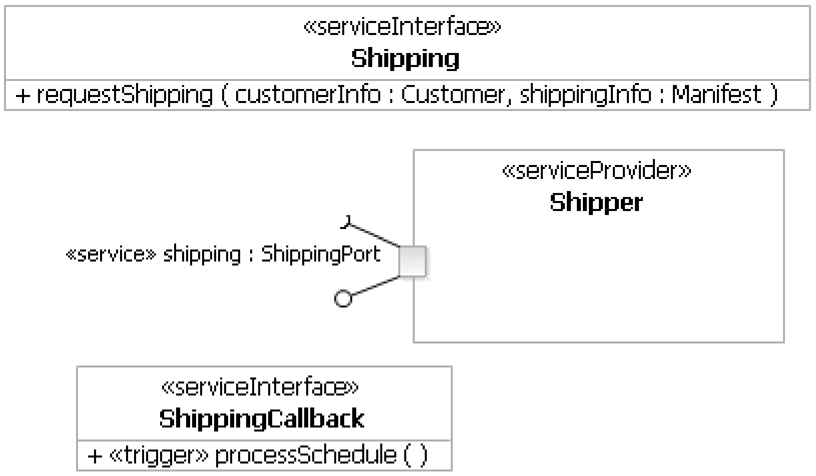
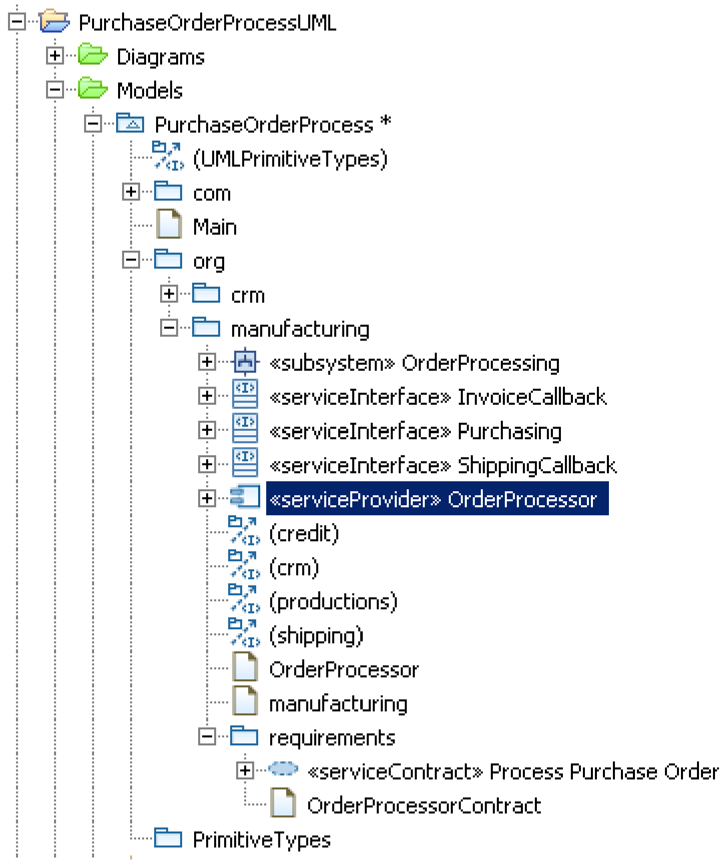

«serviceInterface»
Shipping
+ requestShipping (customerInfo : Customer, shippingInfo : Manifest)
«serviceProvider»
Shipper
«service» shipping : ShippingPort
«serviceInterface»
ShippingCallback
+ «trigger» processSchedule ( )

Figure 10: The Shipper Service Provider

C.4.5 Service Model

The purchase order processing services are specified by the Purchasing interface, and provided by the OrderProcessor component. This component provides the Purchasing interface through its purchasing port. All customer interaction is through this port separating customer clients from interactions the component may have with other service consumers or providers. This limits coupling in the model making it easier to change as the market and service consumers and providers change.

The organization of the OrderProcessor component is shown in the Project Explorer view below.



PurchaseOrderProcessUML
Diagrams
Models
PurchaseOrderProcess *
(UMLPrimitiveTypes)
com
Main
org
crm
manufacturing
«subsystem» OrderProcessing
«serviceInterface» InvoiceCallback
«serviceInterface» Purchasing
«serviceInterface» ShippingCallback
«serviceProvider» OrderProcessor
(credit)
(crm)
(productions)
(shipping)
OrderProcessor
manufacturing
requirements
«serviceContract» Process Purchase Order
OrderProcessorContract
PrimitiveTypes

Figure 11: The manufacturing Business Functional Area (Package)

The OrderProcessor service provider is contained in the org::manufacturing package which is used to organize services related to manufacturing. The manufacturing package also contains all the interfaces provided by the components contained in the package.

The OrderProcessor diagram below provides an external view of the OrderProcessor service provider and its provided and required interfaces. The external or “black-box” view is what clients of, and collaborators with the service provider see. The component’s internal structure shown later in the example provides an internal or “white-box” view of the structure supporting the component’s implementation. Note that the external view is not a specification separate from an implementation, it is simply a view of some aspects of the component. If the architect or developer wanted to completely separate the specification of the service provider from its possible implementations, a service specification would be used. A service specification defines a contract between service consumers and service providers that decouples them from particular provider implementations. The specification component can be realized by perhaps many concrete components that actually provided the services in a manner that fulfills the contract and provides acceptable qualities of service.

Note that the contract specified by a service specification is quite different than the service contract corresponding to the business operational requirements. They both capture similar information about interfaces, operations, structure, and behavior, but the service contract represents an architecturally neutral requirements contract that is fulfilled in some way. A service specification is an architecturally specific contract that is realized in some way. The difference is in the flexibility of fulfillment (through a CollaborationUse and role bindings) and the more semantically constrained realization.

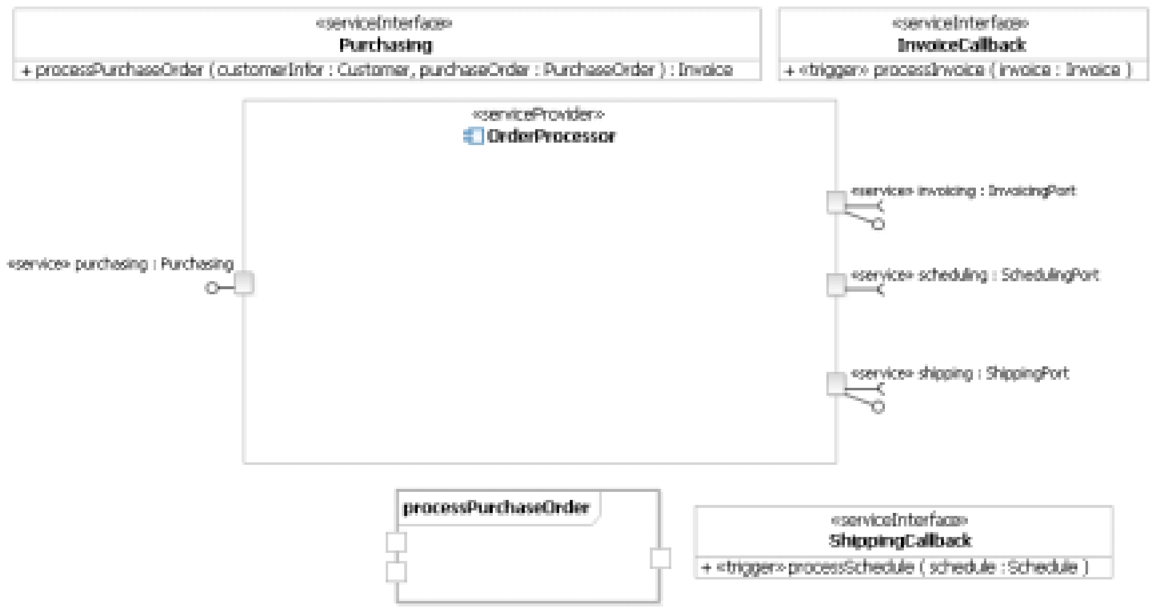

«serviceInterface»
Purchasing
+ processPurchaseOrder (customerInfor : Customer, purchaseOrder : PurchaseOrder) : Invoice
«serviceInterface»
InvoiceCallback
+ «trigger» processInvoice (invoice : Invoice)
«serviceProvider»
OrderProcessor
«service» invoicing : InvoicingPort
«service» purchasing : Purchasing
«service» scheduling : SchedulingPort
«service» shipping : ShippingPort
processPurchaseOrder
«serviceInterface»
ShippingCallback
+ «trigger» processSchedule (schedule : Schedule)

Figure 12: The OrderProcessor Service Provider

The OrderProcessor service provider component is sufficiently simple, and stable that the architect/developer decided not to use a service specification. As a result, any service consumer using the OrderProcessor component will be coupled to this particular implementation. Whether this is a sufficient design depends on how much the service will be used and how much it might change over time. Only the solution architect can decide what level of coupling is tolerable for a particular system.

The OrderProcessor component also has service interaction points to interact with three other service providers: invoicing, scheduling and shipping. These service providers provide services used by the OrderProcessor component in order to implement its provided service operations.

Each of these service interaction points is involved in a simple protocol involving provided and required interfaces. For example, the invoicing interaction point requires the Invoicing interface to initiate price calculations and send the shipping price. However, it may take some time to calculate the shipping price, so the OrderProcessor provides the InvoiceCallback interface through its invoicing port so the invoicing service provider can send an invoice when it is ready.

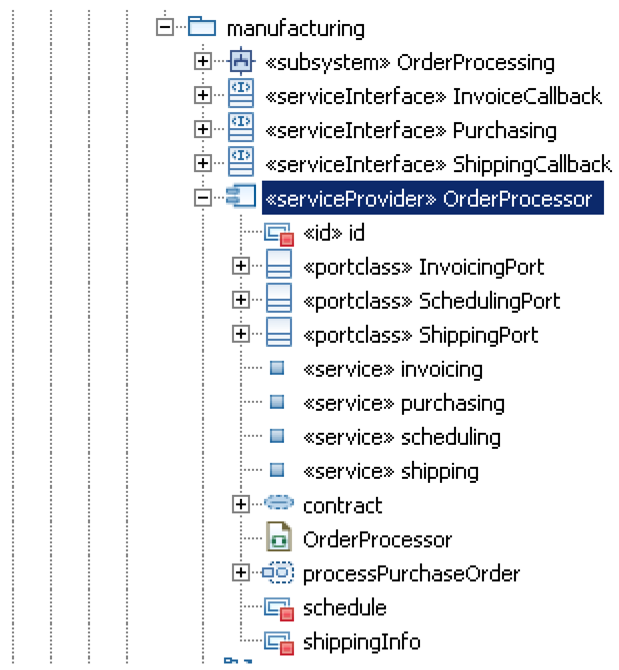
Notice that there is a potential design problem with this example resulting in undesirable coupling. The rules for how an order processing service provider interacts with invoicing are captured in the same business process (Activity) as the rules for interfacing with production scheduling and shipping service providers. This makes it difficult to reuse invoicing, scheduling, and/or shipping services without duplicating the interaction protocols. This coupling effect often results from a direct implementation of business processes as service operations. Business process models focus on the steps in an activity necessary to accomplish some specific business goal such as efficient order processing. They do not necessarily think about how to refactor those steps to increase cohesion, decrease coupling, enable reuse, minimize distributed communication overhead, handle network security, manage integrity of persistent data, etc. These are all IT concerns that must be addressed using software architectures and practiced and proven design patterns. Using service contracts provides a way to formally capture business requirements while allowing refactoring into service providers using architectures that meet the business requirements and also address IT concerns. The linkage between the architected solutions and the business requirements are through contract fulfillment which bind the parts of the service providers with the roles they play in the service contract. We don’t address these design problems further in this simple example.

C.4.6 processPurchaseOrder Implementation Model

We have now completed the architectural structure of the service model and capture it in external views of service providers. The next thing to do is to design an implementation for each service operation provided by each service provider. These implementations must conform to any fulfilled service contracts or realized service specifications. We will focus on the OrderProcessor service provider and its processPurchaseOrder service operation.

C.4.6.1 Internal Structure

The OrderProcessor service provider provides a single service interface, Purchasing through its purchasing service interaction point. This service interface specifies a single operation, processPurchaseOrder. A service provider must provide some implementation for all of its provided service operations. This example uses an Activity to model the implementation of the processPurchaseOrder service operation. The details for how this is done are shown in the internal structure of the OrderProcessor component providing the service. The OrderProcessor internal structure is captured in diagrams, interfaces, classes and activities contained in the component as shown in the Project Explorer view below.



- manufacturing
  - «subsystem» OrderProcessing
  - «serviceInterface» InvoiceCallback
  - «serviceInterface» Purchasing
  - «serviceInterface» ShippingCallback
  - «serviceProvider» OrderProcessor
    - «id» id
    - «portclass» InvoicingPort
    - «portclass» SchedulingPort
    - «portclass» ShippingPort
    - «service» invoicing
    - «service» purchasing
    - «service» scheduling
    - «service» shipping
    - contract
    - OrderProcessor
    - processPurchaseOrder
    - schedule
    - shippingInfo

Figure 13: The Organization of the OrderProcessor Service Provider

The OrderProcessor composite structure diagram provides an overview of the service provider's internal structure. This shows the parts (interaction points and properties) that make up the static structure of the component. The convention used in this example is to use a class diagram whose name is the same as the component, and in the package containing the component, to show its external view. A composite structure diagram of the same name as the component and contained in the component provides the internal view of the service provider's structure. This makes it easy to coordinate the external view and internal view of the service provider, and scope the diagrams the same as the component.

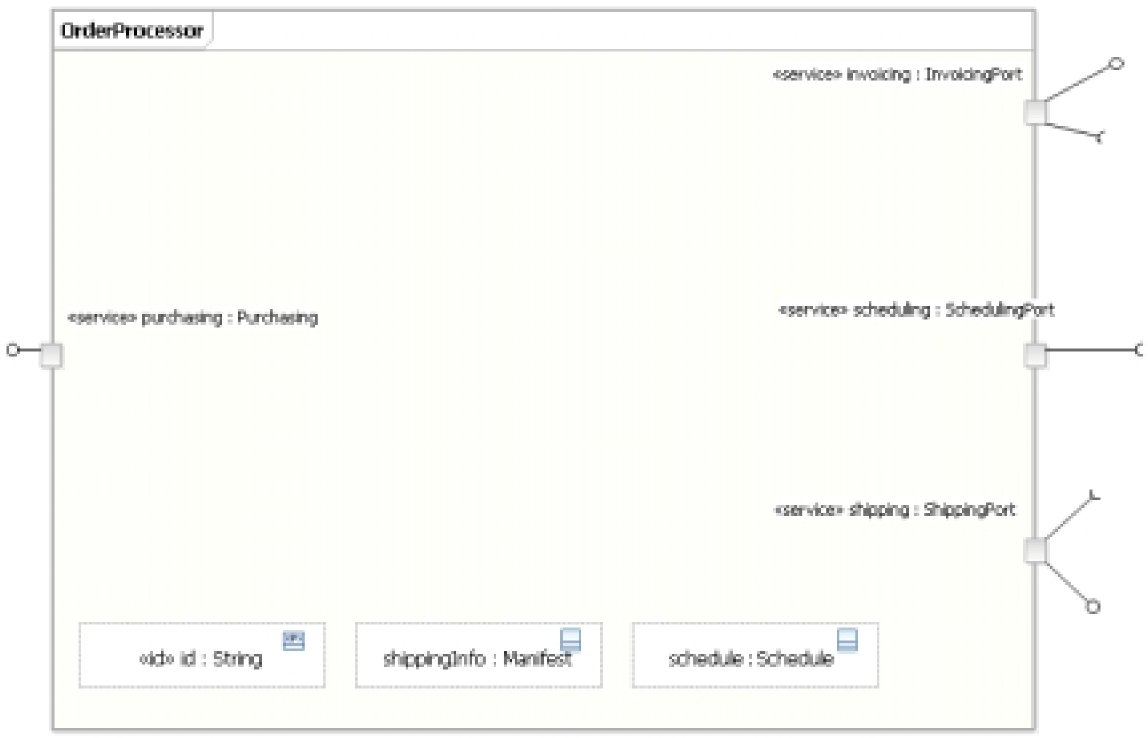

OrderProcessor
«service» invoicing : InvoicingPort
«service» purchasing : Purchasing
«service» scheduling : SchedulingPort
«service» shipping : ShippingPort
«id» id : String
shippingInfo : Manifest
schedule : Schedule

Figure 14: The Internal Structure of the OrderProcessor Service Provider

The internal structure of the OrderProcessor component is quite simple. It consists of the service interaction points for the provided and required interfaces plus an number of other properties that maintain the state of the service provider. The id property is used to identify instances of the service provider. This property will be used to correlate consumer and provider interaction at runtime. The schedule and shippingInfo properties are information used in the implementation of the processPurchaseOrder service operation.

The types of the invoicing, scheduling and shipping ports are the InvoicingPort, SchedulingPort a ShippingPort «portclass» classes respectively. These classes are necessary because a service interaction point is a typed element and its provided and required interfaces are derived from the interfaces realized and used by its type classifier. These classes have little logical use in the model and should be automatically maintained as needed by tools. The exception is the purchasing service interaction whose type is the Purchasing interface. If the type of an interaction point is an interface, then the interaction point provides that interface.

C.4.6.2 Implementing Provided Service Operations

Each service operation provided by a service provider must be realized by either:

1. a Behavior (Activity, Interaction, StateMachine, or OpaqueBehavior) that is the method of the Operation, or
2. an AcceptEventAction (for asynchronous calls) or AcceptCallAction (for synchronous request/reply calls) in some Activity belonging to the component.

> This allows a single Activity to have more than one (generally) concurrent entry point and corresponds to multiple receive activities in BPEL. These AcceptEventActions are usually used to handle callbacks for returning information from other asynchronous CallOperationActions.

The OrderProcessor component has an example of both styles of service realization. The processPurchaseOrder operation has a method Activity of the same name which is an owned behavior of the service provider providing the service operation.

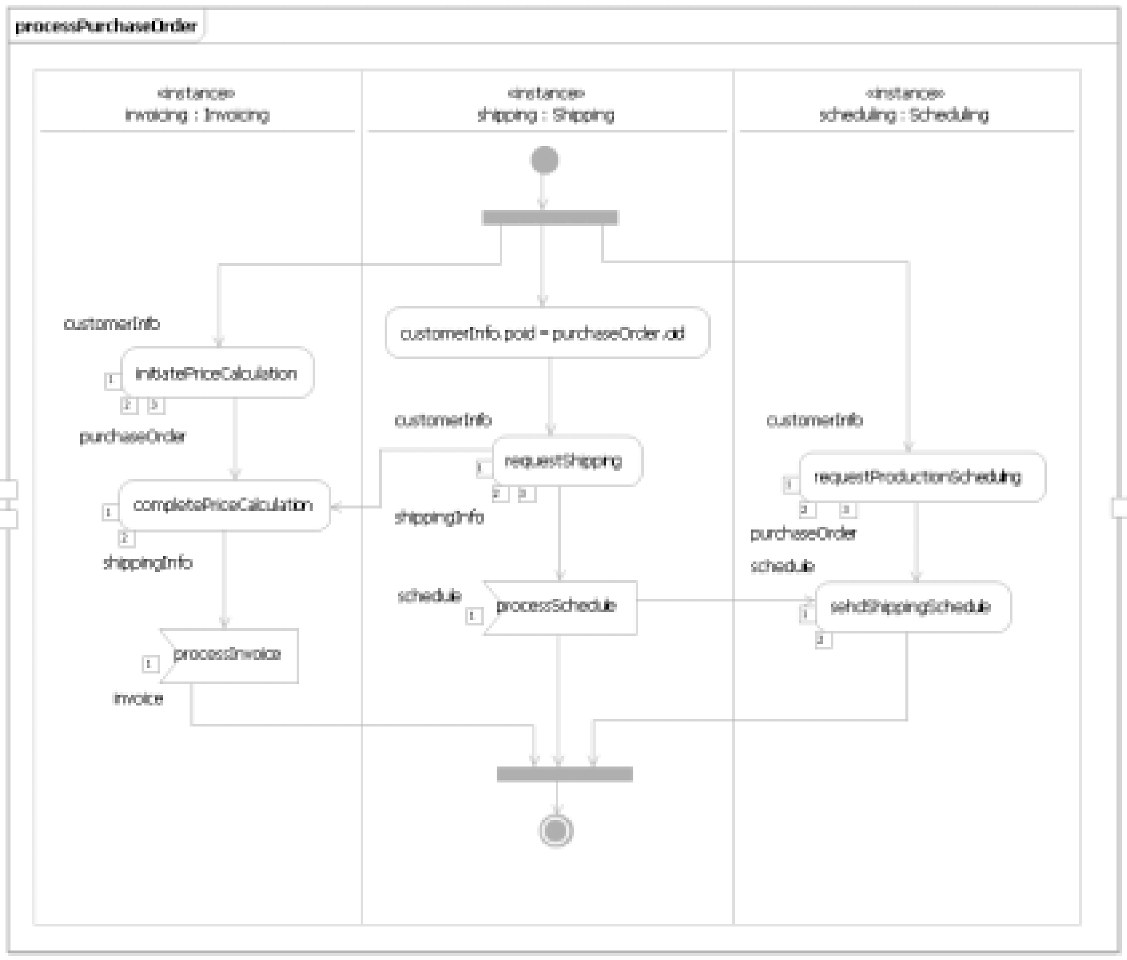


Figure 14: The processPurchaseOrder Service Operation Implementation

This diagram corresponds very closely to the BPMN diagram and BPEL process for the same behavior. The InvoiceCallback and ShippingCallback service operations are realized through the processInvoice and processSchedule accept event actions in the process. Notice that the corresponding operations in the interfaces are denoted as «trigger» operations to indicate the ability to respond to AcceptCallActions (similar to receptions and AcceptEventActions where the trigger is a SignalEvent. The «trigger» keyword is not currently part of UML2. An issue has been raised.

C.4.6.3 Fulfilling Service Contracts

The OrderProcessor component is now complete. But there are two things left to do. First we need to tie the OrderProcessor service provider back to the business process that modeled its requirements. Then we need to create subsystem that connects service providers capable of providing the OrderProcessor's required interfaces to the appropriate service interaction points. This will result in a deployable subsystem that is capable of executing. This section will deal with linking the SOA solution back to the business requirements. The next section covers the deployable subsystem.

Note that nothing in this section will change anything in how the OrderProcessor component is translated to a SOA implementation. Linking components to services contract only describes how the component fulfills the requirements specified by those contracts. This does not effect the service provider's implementation or how it would be translated to a SOA platform. However, the linkage is more complex than a simple dependency. It shows specifically what parts of the service provider play the roles in the service contract, and how constraints on the component fulfill business constraints.

Section 1.3 described the requirements for the OrderProcessor service provider using a service contract that provided a role-centered view of the business processes created by a business analyst. A collaboration use can be added to the OrderProcessor service provider to indicate the service contracts its fulfills.

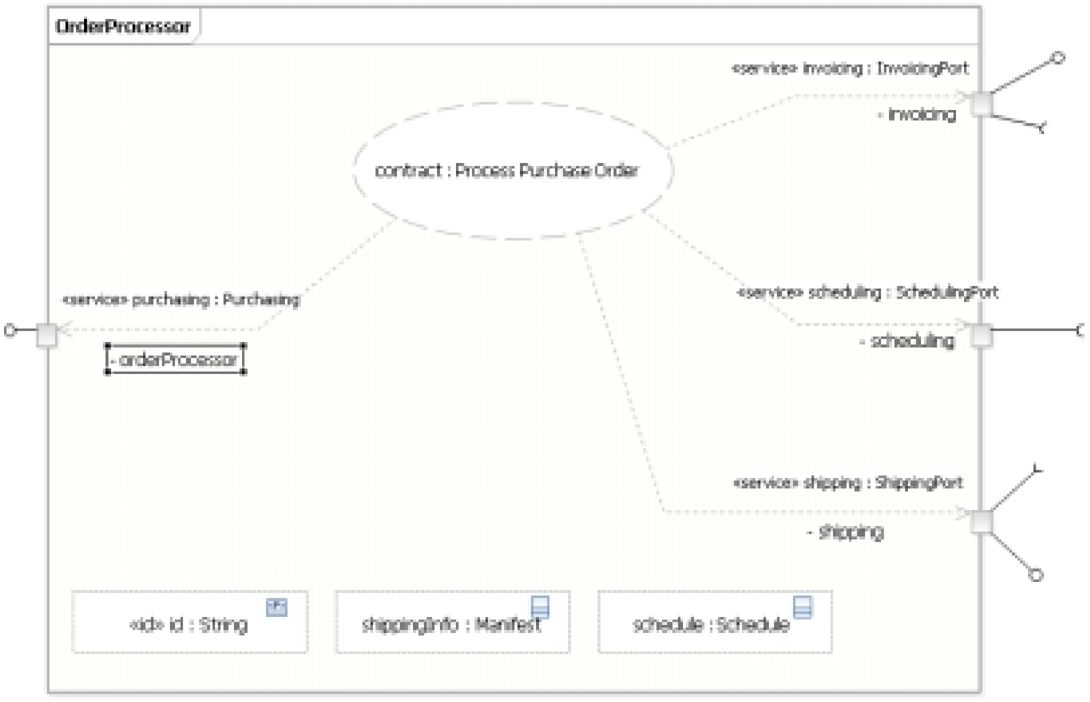

OrderProcessor
«service» invoicing : InvoicingPort
- invoicing
contract : Process Purchase Order
«service» purchasing : Purchasing
- orderProcessor
«service» scheduling : SchedulingPort
- scheduling
«service» shipping : ShippingPort
- shipping
«id» id : String
shippingInfo : Manifest
schedule : Schedule

Figure 15: Fulfilling the Service Contract

The collaboration use, called contract, is an instance of the Purchase Order Process service contract. This specifies the OrderProcessor service profiler fulfills the Purchase

Order Process services. The role bindings indicate which part of the service provider plays the role in the service contract. For example, the invoicing port plays the Invoicer role. The purchasing port plays the orderProcessor role.

These role bindings are not related to the service channel connectors described in the next section. Connectors are used to connect parts in a subsystem that provide required interfaces. Role bindings specify what role the part plays in a service contract. The interpretation of a role binding can be either strict or loose. Strict contract fulfillment means the parts are type compatible with the roles they are bound to. Loose contract fulfillment means the parts are intended to play those roles according to the architect, but the tools need not, and perhaps could not verify role and part compatibility. This is perhaps because the business service contract is incomplete or an informal sketch.

C.4.6.4 Implementing the OrderProcessing Subsystem

Recall the manufacturing package also contained an OrderProcessing subsystem.

This subsystem represents a deployable component that connects the OrderProcessor service provider with other service providers that provide its required interfaces. This subsystem is an assembly of components that provide all the information necessary to deploy and execute the OrderProcessor services.

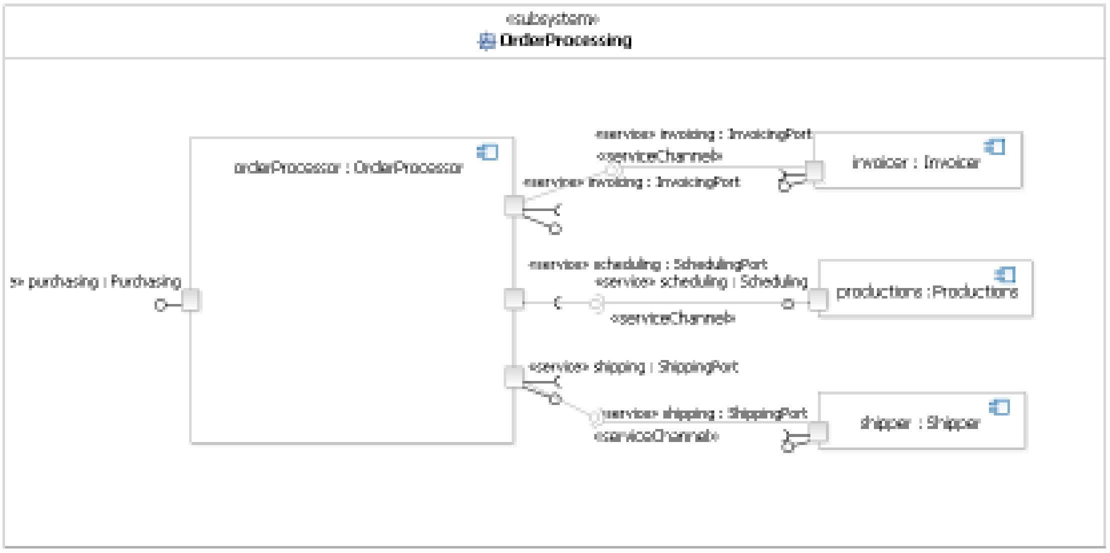

«subsystem»
OrderProcessing
orderProcessor : OrderProcessor
«service» invoicing : InvoicingPort
«serviceChannel»
«service» invoicing : InvoicingPort
invoicer : Invoicer
«service» scheduling : SchedulingPort
«service» scheduling : Scheduling
productions : Productions
«serviceChannel»
»» purchasing : Purchasing
«service» shipping : ShippingPort
«service» shipping : ShippingPort
«serviceChannel»
shipper : Shipper

Figure 16: Assembling the Parts into a Deployable Subsystem

The OrderProcessing subsystem consists of an instance of the OrderProcessor, Invoicer, Productions, and Shipper components. The invoicing service of the orderProcessor component is connected to the invoicing service of the invoicer component. The orderProcessor component requires the Invoicing interface which is provided by the invoicer service provider. It also provides the Scheduling service provider for the invoicer in order to receive the updated schedule information. The connected services can offer different binding styles. The service channel between the service interaction points can specify the actual binding style to use.

The other services are similarly connected to providers.

The OrderProcessing subsystem is now complete and ready to be deployed. It has specific instances of all required service providers necessary to fully implement the processPurchaseOrder service. Once deployed, other service consumers can bind to the order processor component and invoke the service operation.