

METEOR-S PROCESS DESIGN AND DEVELOPMENT TOOL

by

RANJIT MULYE

(Under the Direction of John A. Miller)

ABSTRACT

The growing popularity of Service Oriented Computing based on Web services standards is creating a need for paradigms to represent and design business processes. Significant work has been done in the representation aspects with regards to WSBPEL. However, design and modeling of business processes is still an open issue. In this thesis, we present a novel designer which supports intuitive modeling of Web processes, using a template based approach for semi-automatically integrating partners either at design time or at deployment time. The process design and development tool described in this work helps process developers create complex business processes using an easy to use Graphical User Interface. The tool which provides a variety of features is developed as a plugin to Eclipse platform. This work has been done as part of the METEOR-S project at the Large Scale Distributed Information Systems (LSDIS) lab at The University of Georgia.

INDEX WORDS: WSDL, WSBPEL, Business Process Design, Process Modeling,
METEOR-S, Semantic Web services

METEOR-S PROCESS DESIGN AND DEVELOPMENT TOOL

by

RANJIT MULYE

B.E., Pune University, Pune, India, 2000

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2004

© 2005

Ranjit Mulye

All Rights Reserved

METEOR-S PROCESS DESIGN AND DEVELOPMENT TOOL

by

RANJIT MULYE

Major Professor: John A. Miller

Committee: Amit P. Sheth

Maria Hybinette

DEDICATION

To Aai and Baba

ACKNOWLEDGEMENTS

I would like to thank my Major Advisor, Dr. John A. Miller for all the support and guidance he extended to me throughout this work. I would also like to thank Dr. Amit P. Sheth for his valuable suggestions, encouragement and support that helped me with my thesis. Special thanks, to Dr. Maria Hybinette for providing me with guidance, help and assurance in difficult times during my thesis work. I would also like to acknowledge the help given by my fellow-students of the METEOR-S project group. Brains storming sessions with Kunal and his suggestions have been really useful. Karthik's effort in testing and ensuring of proper working of the tool came as a great help. This work was done as part of the METEOR-S project at the LSDIS lab. I am thankful to Dr. Sheth for providing with giving me the opportunity to use lab resources and work in the intellectual environment. I would like to thank my sister and brother-in-law, for their continuous encouragement and love. Finally, I sincerely thank my parents for their support and the sacrifices they made to get me into graduate school. Words are inadequate to express my gratitude to them for giving me the courage and strength I needed to complete my goals especially during the tension-filled, sleepless days.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	v
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTERS	
1. INTRODUCTION	1
2. BACKGROUND	6
3. METEOR-S PROCESS DESIGNER TOOL.....	19
4. FEATURES OF THE METEOR-S PROCESS DESIGN TOOL.....	33
5. RELATED WORK	39
6. USE CASE.....	42
7. CONCLUSION AND FUTURE WORK	48
REFERENCES	51
APPENDICES	
APPENDIX A: INSTALLATION GUIDE	57
APPENDIX B: USERS GUIDE.....	59

LIST OF TABLES

	Page
Table 1: Comparison of UML and WSBPEL impressibility	11
Table 2: Activities that belong to the Definition group	25
Table 3: Activities that belong to the Basic activity group.....	26
Table 4: Activities that belong to the structured activity group.....	27
Table 5: Extensional group	28
Table 6: Packages and tools used.....	31
Table 7: Color codes for activity groups.....	36
Table 8: Comparison of Web process modeling approaches.....	40
Table 9: Comparison of BPEL designing Tools	41

LIST OF FIGURES

	Page
Figure 1: Web Service Composition.....	7
Figure 2: Web Service Orchestration.....	8
Figure 3: Web Service Choreography.....	9
Figure 4: METEOR-S System Architecture	14
Figure 5: Semantic Template annotated with Ontology Concepts	18
Figure 6: METEOR-S Process design and development tool.....	20
Figure 7: Model-View-Controller Architecture.....	22
Figure 8: METEOR-S Semantic Process design and development tool Architecture.....	23
Figure 9: UML diagram for model class, Part I.....	29
Figure 10: UML diagram for model class, Part II	30
Figure 11: Phase II – Dynamic partner discovery	35
Figure 12: Definition lookup	37
Figure 13: Status bar helper messages	38
Figure 14: Steps for purchase order process	43
Figure 15: Use Case – Purchase Order Process	44
Figure 16: Use Case – Purchase Order BPEL	46
Figure 17: Use Case – Purchase Order Process WSDL.....	47
Figure 18: Starting the BPEL Designer project	60
Figure 19: Creating a new BPEL process File.....	61
Figure 20: Skeleton BPEL Process.....	62

Figure 21: Sample propertySheet.....	63
Figure 22: Sample process WSDL.....	65

CHAPTER 1

INTRODUCTION

Many businesses are adopting Web service technology to expose their business applications, allowing them to have business collaboration both, within their organization and with business partners outside the organization. Adoption of the Service Oriented Architecture (SOA) helps businesses contract-out their non-critical functions. Wide acceptance of Web services is largely due to the fact that they are built on XML based standards like SOAP, WSDL and UDDI. Simple Object Access Protocol (SOAP) [44] is a lightweight protocol for exchange of information among Web Services. Web Service Description Language (WSDL) [13] describes a Web service. The Universal Description, Discovery and Integration (UDDI) [1] protocol creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web services over the Internet. It specifies the location of the service and the operations (or methods) the service exposes.

The new world economy calls for business processes that span different organizational units as well as across organizations themselves. These different units or organizations as a whole typically have heterogeneous platforms and systems. Web Services are inherently designed for interaction in a loosely coupled environment. This makes Web Services an ideal choice for companies that expect to have inter-departmental or inter-organizational interactions. With the increasing popularity of Web Services, a new paradigm of business processes is gaining significant attention. The services offered by various businesses can be interconnected into

complex business processes or workflows. The Web Services Business Process Execution Language (WSBPEL) [1] provides a specification of business processes and business interaction protocols. WSBPEL defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners. . WSBPEL defines an interoperable integration model to facilitate intra-corporate and business-to-business process integration. The idea of building such types of combined processes has brought importance for tools for modeling of business processes for Web service orchestration.

Though languages like WSBPEL offer solutions for integrating Web Services into a business process, they are difficult to learn and often involve understanding complex language syntax. With the need to build complex business process, tools that would aid process developers in building such processes are gaining importance. Along with providing ease of process creation, tools are expected to be smarter and more sophisticated. In this work, we present such a WSBPEL Web process designer tool – METEOR-S Process design and development tool. The METEOR-S Process design and development tool offers a GUI based design interface to allow business process design.

The METEOR-S [3] research project at the Large Scale Distributed Information Systems Lab, at the University of Georgia, is an effort towards developing a system to support development of a complete business processes lifecycle and to leverage the use of semantics in each step of the development cycle. The Process design and development tool discussed in this effort is part of the METEOR-S project. It offers an easy to use graphical user interface for building business processes targeting for the WSBPEL process definition languages. The Process design and development tool also offers functionality for dynamic discovery of Web Service partners to optimize the process.

The term Web process as used in the METEOR-S project represents realization of a business process over the Web infrastructure using the Web services as components. A key element of any Web process is the partners that the process interacts with. Currently, the partner selection process is mostly static, wherein partners are either selected ahead of time or the process developer manually connects to a UDDI registry [1] and tries to search for services that fit his requirements. Though this approach would work for many traditional business processes, it does not guarantee that a partner that would be used in the resulting business process would be necessarily optimal. This is so because the partner selection process is static and often there is a significant time gap between the partner selection process and their incorporation into a particular process. It is possible that services that can offer better results in term of various quality parameters (costs, time, reliability, accuracy, etc.) may have been deployed after the partners for the current process were finalized. Also, it is possible that the service may no longer be accessible and the process developer would not be aware of the same.

Consider a scenario wherein a process developer is designing a process to execute a purchase order. The process is invoked by a purchase order request sent by a partner (buyer), specifying the desired quantity of items s/he plans to purchase. This is followed by querying the supplier (partner) for availability of items. If the desired quantity of the order item is available in the inventory, then the order is placed with the supplier service. Upon successful completion of the order, an invoice is generated for the purchase order and returned to the caller of the process (buyer). Generally, the order is fulfilled by procuring goods from various suppliers. However, using static partner selection, the business process would be tied to a single supplier. This would mean that the process developer would have to write separate processes for each supplier with whom s/he wishes to interact with. Also, this approach limits the amount of dynamic selection

the process developer can achieve in selecting the supplier to choose. A better approach would be either to dynamically select a partner during the design phase of the process or assume a virtual supplier while designing the process and perform partner discovery at deployment-time to achieve optimal benefits. Using this approach, the process developer can write a single process that can act as a blue print and may deploy multiple instances of the process by selecting a dynamic partner depending on varying orders or needs.

Secondly, since most of the partner searches are keyword-based, the process developer may not be aware of similar services that offer a similar functionality just because they are differently named. Further, most of the registries offer searching only on business names, service names or T-models. The naming conventions adopted by companies to name their businesses or services may vary and the T-model names would be set by specific service implementers. These naming conventions may not be well suited to search for the desired functionality.

The METEOR-S Process design and development tool uses semantic template based partner discovery to realize the above mentioned functionality. Using our tool, the process developer would be able to semantically describe what s/he expects from the partner service and perform partner selection at design time or defer it till deployment time. Using semantics in describing Web Services and discovering them helps overcome the limitations faced due to conventional keyword based discovery described above. The Semantic Template is based on the WSDL-S [21] specification which is an extension to the (WSDL) specification which supports semantic annotation of Service elements. A new version of WSDL-S is being jointly proposed by LSDIS lab and IBM research.

The METEOR-S Process design and development tool helps Web Process Developers build complex processes using the WSBPEL specifications. It lets a process designer stress on the logic of the process rather than be overwhelmed by the syntax of the specification language. As mentioned earlier, the process generates an executable BPEL Web Process file and a corresponding process WSDL. The tool has been tested to create Web Processes and deploy and run them on freely available BPEL engines, viz. BPWS4J [40] and ActiveBPEL [41].

CHAPTER 2

BACKGROUND

In this chapter, we present an overview of the present composition standards and a way to model business processes. We then present different proposed ways in which Web Processes are realized and try to justify the choice of a GUI based design approach for the METEOR-S Process design and development tool. Later, we describe Semantic Web services and their significance. Finally, we conclude this chapter describing the METEOR-S project in detail.

2.1 WEB SERVICE COMPOSITION, ORCHESTRATION AND CHOREOGRAPHY

While many companies have begun to deploy individual Web services, the real value will come when enterprises can connect services together, providing higher value to an organization. Connecting various Web services together has been termed differently according to the perspective of the designer. While somewhat overlapping, the two terms orchestration and choreography describe two aspects of creating business processes from composite Web services. A detailed discussion about the differences in approach of Choreography and Orchestration are discussed in [45].

2.1.1. Web Service Composition

The term Service Composition refers to modeling and execution of Web processes with individual Web services as its component. The goal is to use Web Services as individual building blocks to build new (Composite) Services. Figure 1 gives an overview of a typical Web service composition.

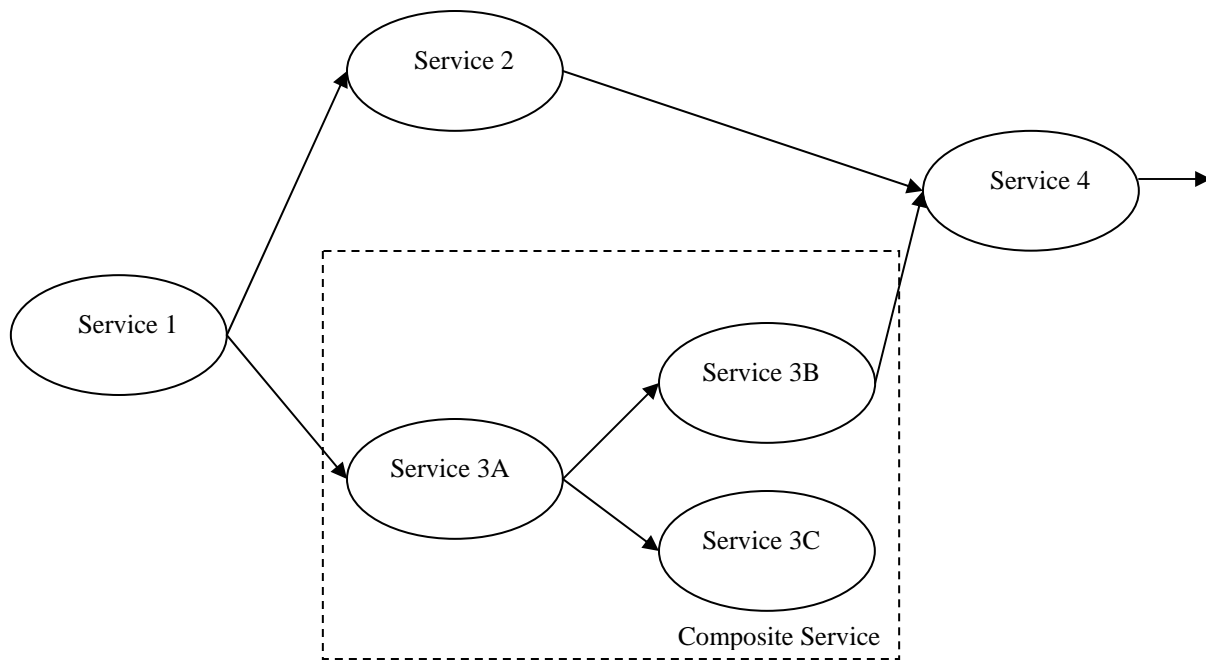


Figure 1: Web Service Composition

2.1.2. Web Service Orchestration

Orchestration of Web Services refers to defining an executable business process interacting with Web services within and outside the organizational boundary. Orchestration is characterized by the following features:

- Defines the sequence and conditions in which one Web Service invokes other services in order to achieve a specific goal.
- Defines a pattern of interactions that the partner services must follow to realize the expected functionality.
- Characterized by a central Service which acts a controller for other partner services.

Figure 2 illustrates an example of Service Orchestration. As seen in the illustration, the main partner service is in control of the entire process execution. The main partner decides the flow of

the process and makes decisions on the order of invocation or control. In short, orchestration is a more detailed, execution-driven mechanism, viewed from the perspective of routing a particular set of messages through a process. The WSPBEL standard provides specification for achieving Service Orchestration.

The METEOR-S Process design and development tool follows the Web Service Orchestration methodology to build executable composite Web Services.

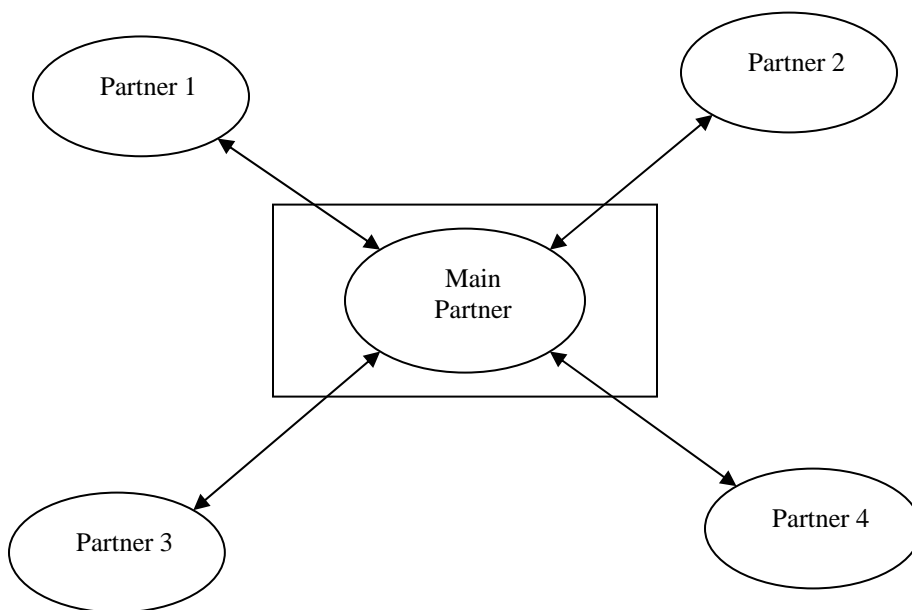


Figure 2: Web Service Orchestration

2.1.3. Web Service Choreography

Web services choreography deals with the interactions of services with other partner services. It defines a model for the sequences of operations, states, and conditions, to achieve a specific purpose or goal. Orchestration is characterized by the following features:

- Collaborative in nature, where each participating Service in the process describes the part it plays in the interaction.
- Associated with the public message exchanges that occur between multiple Web services.
- Abstract and descriptive in nature and viewed from the perspective of the parties that exchange messages to accomplish a particular process.

The World Wide Web Consortium (W3C) has issued the Web Services Choreography Description Language Version 1.0 as a W3C First Public Working Draft. Figure 3 illustrates service interaction using Service Choreography.

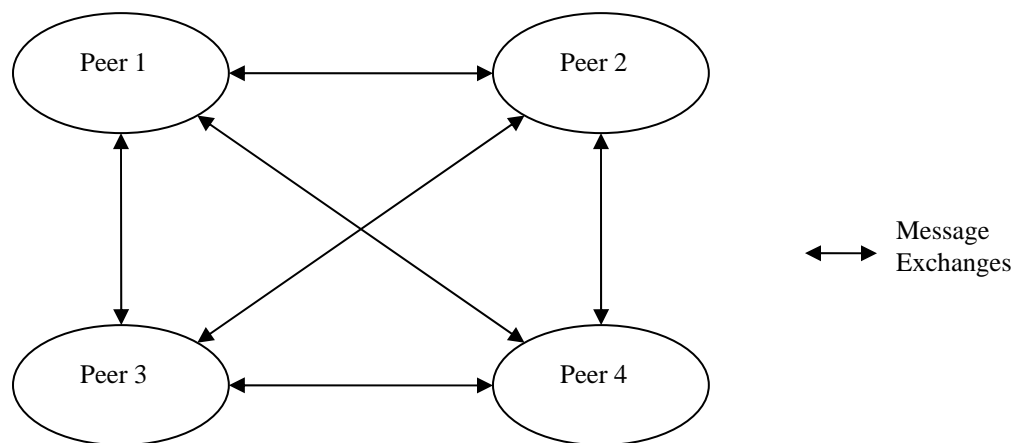


Figure 3: Web Service Choreography

2.2. WEB PROCESS MODELING

The METEOR-S Process design and development tool generates executable Web process using the WSBPEL specification. Different approaches have been proposed to model a WSBPEL process. These approaches help design the process and later export the model in WSBPEL

format. To understand the effectiveness of various Web process modeling specification, it is first necessary to find the common patterns used in Web process analysis. Following is a list of the relevant patterns of a Web process and their grouping.

- **Basic Control**

Sequence, Parallel Split, Synchronization, Exclusive Choice, Simple Merge

- **Advanced Branching and Synchronization**

Multi-Choice, Synchronization Merge, Multi-merge, Discriminator,

- **Structured Patterns**

Arbitrary Cycles, Implicit Termination

- **Patterns involving Multiple Instances**

Multiple Instances Without Synchronization, Multiple Instances With a Priori Design Time Knowledge, Multiple Instances With a Priori Runtime Knowledge, Multiple Instances Without a Priori Runtime Knowledge

- **State-based Patterns**

Deferred Choice, Interleaved Parallel Routing, Milestone

- **Cancellation Patterns**

Cancel Activity, Cancel Case

A comprehensive description and analysis of all the above patterns is given in [6].

Often UML is proposed for modeling of business process and later mapping the UML constructs to WSBPEL process elements. Table 1 gives a summary of process patterns that can be modeled using UML Activity Diagrams and using WSBPEL.

Table 1: Comparison of UML and WSBPEL impressibility*Source [6]*

Pattern	Standard	
	UML Activity Diagrams	WSBPEL
Sequence	+	+
Parallel Split	+	+
Synchronization	+	+
Exclusive Choice	+	+
Simple Merge	+	+
Multi Choice	-	+
Synchronizing Merge	-	+
Multi Merge	-	-
Discriminator	-	-
Arbitrary Cycles	-	-
Implicit Termination	-	+
MI without Synchronization	-	+
MI with a Priori Design Time Knowledge	+	+
MI with a Priori Runtime Knowledge	+	-
MI without a Priori Runtime Knowledge	-	-
Deferred Choice	+	+
Interleaved Parallel Routing	-	+/-
Milestone	-	-
Cancel Activity	+	+
Cancel Case	+	+

As can be seen from Table 1, all the activity patterns supported by WSBPEL cannot be realized using UML constructs, for example Synchronizing Merge or Implicit Termination. The METEOR-S Process design and development tool uses its own abstraction model that resembles the WSBPEL specification. This helps in achieving maximum compatibility with the WSBPEL standards. The modeling approach used by the METEOR-S Process design and development tool is discussed in further details in Chapter 4 which talks about the architecture of the tool.

2.3. BUSINESS PROCESS DESIGN TECHNIQUES

Different methodologies have been suggested for Web process creation. They range from a fully automated generation of Web Processes to manually create Web processes in WSBPEL specification format. Systems such as SWORD [15] and SHOP2 [9] rely on AI planning techniques for creating processes from high level goals. Automated compositions of Web services described in OWL-S [5, 28] process models have been have been proposed. Similarly, automated composition based on Decision-theoretic planning using Markov Decision Processes has been proposed for automated composition [46]. The main goal of such systems is to relieve the process developer of the detailed steps involved in a business process. The process designer is expected to only be able to define the goals of the desired process and the automated tools try to come up with a process that achieves the desired goal. While these systems provide complete automation, such systems have typically been limited to academic environments. This is because businesses are still reluctant to accept a ready made process without any control on the steps involved in the process.

Several tools have been created for semi-automatic composition of Web services [22, 25]. So far, the most successful and popular category of designers has been GUI based designers, that abstract the details of the process language syntax from the user. These tools offer ease of development but at the same time let the process developer decide the exact steps of the process. Some commercially available software [30, 31] offer capabilities to design business process using GUI based approach. The METEOR-S Process design and development tool fits in the GUI based designer category. The Design Tool offers a process designer an easy to use GUI to construct complex processes without getting overwhelmed by the syntax of WSBPEL

specification. It also supports advanced features such as the use of semantic templates to facilitate dynamic partner discovery.

2.4. METEOR-S

As mentioned earlier, the Process design and development tool described in this work constitutes a part of the METEOR-S project at the LSDIS lab at University of Georgia. The METEOR (Managing End-To-End OpeRations) project at the LSDIS lab addresses the issues related to workflow process management for large-scale, complex workflow process applications in real-world multi-enterprise heterogeneous computing environments [8, 42, 43]. The follow-on project, called METEOR-S endeavors to define and support the complete lifecycle of Semantic Web processes [3]. An architectural overview of METEOR-S is presented in figure 4. The key steps in a life cycle of a Web service process are the following:

- Development and Deployment of Semantic Web Services [7, 20, 21]
- Publication and Discovery of Services [23]
- Invocation
- Composition of Web Services [26]

This work focuses on the step of process composition. METEOR-S makes use of semantics in all of the above phases. The different kinds of semantics that METEOR-S tries to exploit are data, functional and Quality of Service semantics. Each type of semantics is used at different levels of the overall process design. A detailed explanation of each is discussed in [22].

For Web services to communicate with each other, they should understand the semantics of each others data. Data semantics tries to associate meaning to the inputs and output of a service helping out in discovery and interoperability. The functional semantics of a Web service operation is a combination of its data semantics, and classification of its operations functionality

as well as its pre-conditions and post-conditions. The Quality of Service (QoS) specifications of a Web service characterize performance and other qualitative/quantitative aspects of Web services. In order for the suppliers of services to understand each others QoS terms, a common understanding must be reached on the meaning of the terms. Ontologies can be used to represent and explicate the semantics of these parameters.

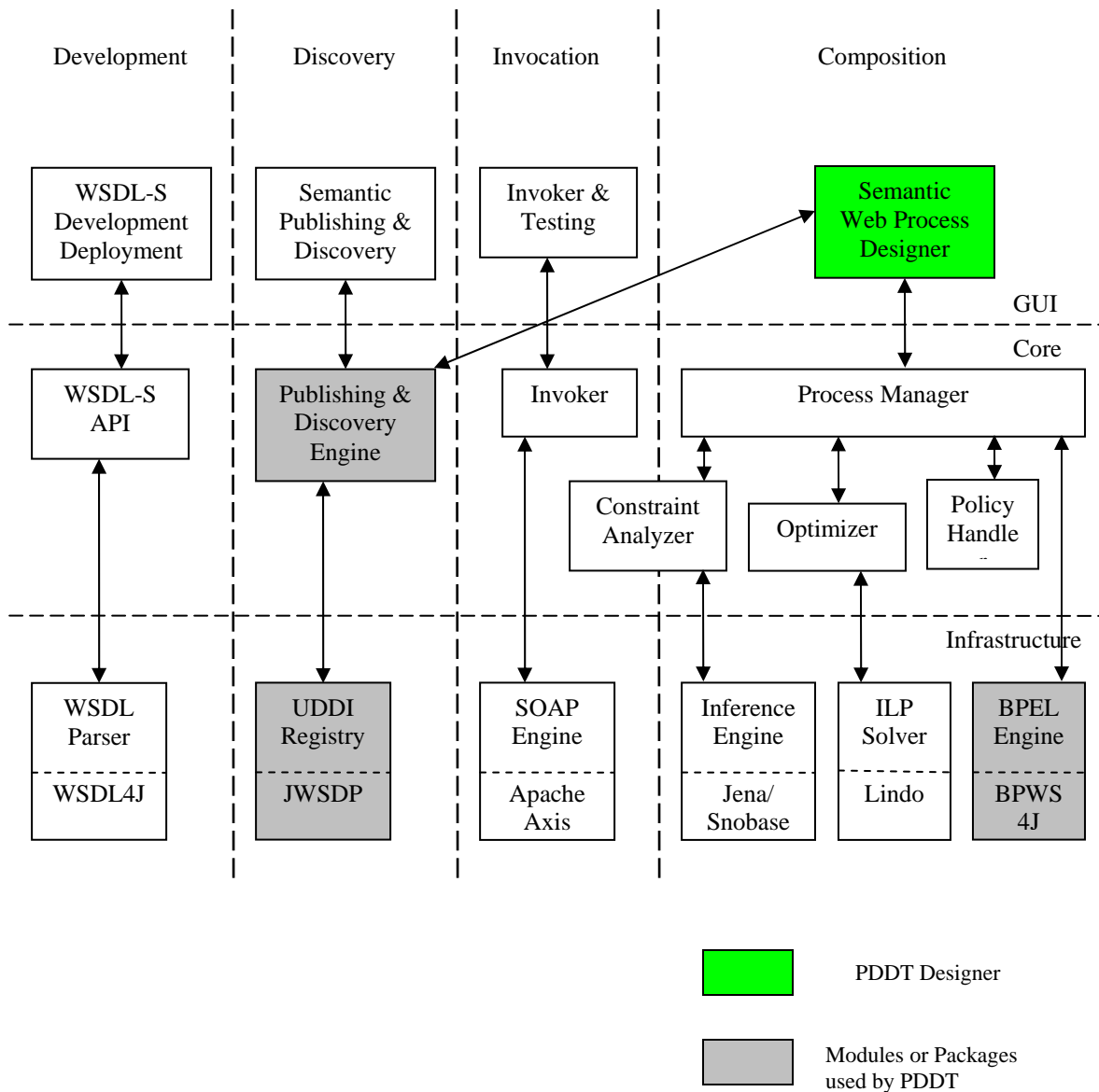


Figure 4: METEOR-S System Architecture

WSDL-S Development & Deployment module helps service developers create annotated WSDL definitions in the form of a new proposed standard, WSDL-S. The Semantic Publishing & Discovery module is used to publish the annotated service definitions to a UDDI registry and later discover them using semantic search mechanisms. Automated invocation and execution of Web Services is handled by the Invoker module. The process manager module is responsible for run-time discovery and optimization of process. As part of the future work, a process developer can perform run-time discovery of partners using semantic templates. The Process Manager uses the semantic discovery engine to search for the required service, based on information present in the service template. The Optimizer and Constraint Analyzer modules discussed in detail in [22] are used to optimize the service results using constraints specified for the service. The Policy Handler engine checks for compatibility of searched services by matching service policy files specified using the WS-Policy specifications [37]. The infrastructure section of the METEOR-S project mainly deals with third party software used by the different modules of the system.

A detailed explanation of the underlying conceptual foundation of METEOR-S is present in [32, 33]. Web Services description by annotation of WSDL using a semi-automatic approach is discussed in [34]. Means of enhancing service description to improve discovery and composition of services is presented in [7]. The METEOR-S Web Services Discovery Infrastructure is explained in detail in [35]. An overview of the Composition Framework of METEOR-S (MWSCF) is presented in [26].

2.5. SEMANTIC WEB SERVICES

One of the issues with search and discovery of Web Services is that currently services are mostly searched by their names. More often than not, due to words taking different meaning in different contexts, it becomes difficult to search for a service that provides a desired operation just by searching on the service name. Also, keyword based search cannot handle the complexities of checking for matching services, in context of service orchestration. A key consideration of a service orchestration is to check for data mapping between outputs and inputs among successive services. These problems can be solved if service properties are described in an interoperable manner.

Considerable work is being applied for utilizing Semantic Web [12] technology to Web Services [16, 19] to achieve the above mentioned goals. The attempt is to describe the semantics of Web Services through the use of ontology languages. The input and output messages of a Web Service can be annotated with concepts from an ontology to describe what they mean. Similarly, operations offered by the service need to be semantically described. Describing semantics of operations is still a research topic and various approaches have been proposed. A straightforward approach is to have a functional ontology that defines actions or verbs. Effort is been put to create functional ontologies for different domains along with proposed languages for the specification of the ontologies [49, 50]. The service operations can then be annotated using the concepts from this functional ontology. Another approach is to define the operation in term of pre-conditions and post-conditions [22]. The METEOR-S process design and development tool both approach. Such annotations would allow search to be based more on “What can this service do functionally?” This would return better results, rather than a search on “What is this

service named”? Also, the use of ontology languages facilitates the service properties to be machine processable in the sense of matching them of their meaning rather than syntax.

This work makes use of the above mentioned data and functional semantics to dynamically search for partners. A process developer can specify the requirements of a partner in the form of a semantic template. The template works as a requirement specification for desired partners. The semantic template consists of a set of operations with their inputs and outputs. All the operations and input/output messages are annotated by concepts from ontologies. A typical semantic template is shown in figure 5 below. The template consists of operations offered by a Web Service and the expected inputs and generated output. These entities are annotated with Ontological concepts. Here, as an example, we use an ontology [48] derived from the RosettaNet [47] standards.

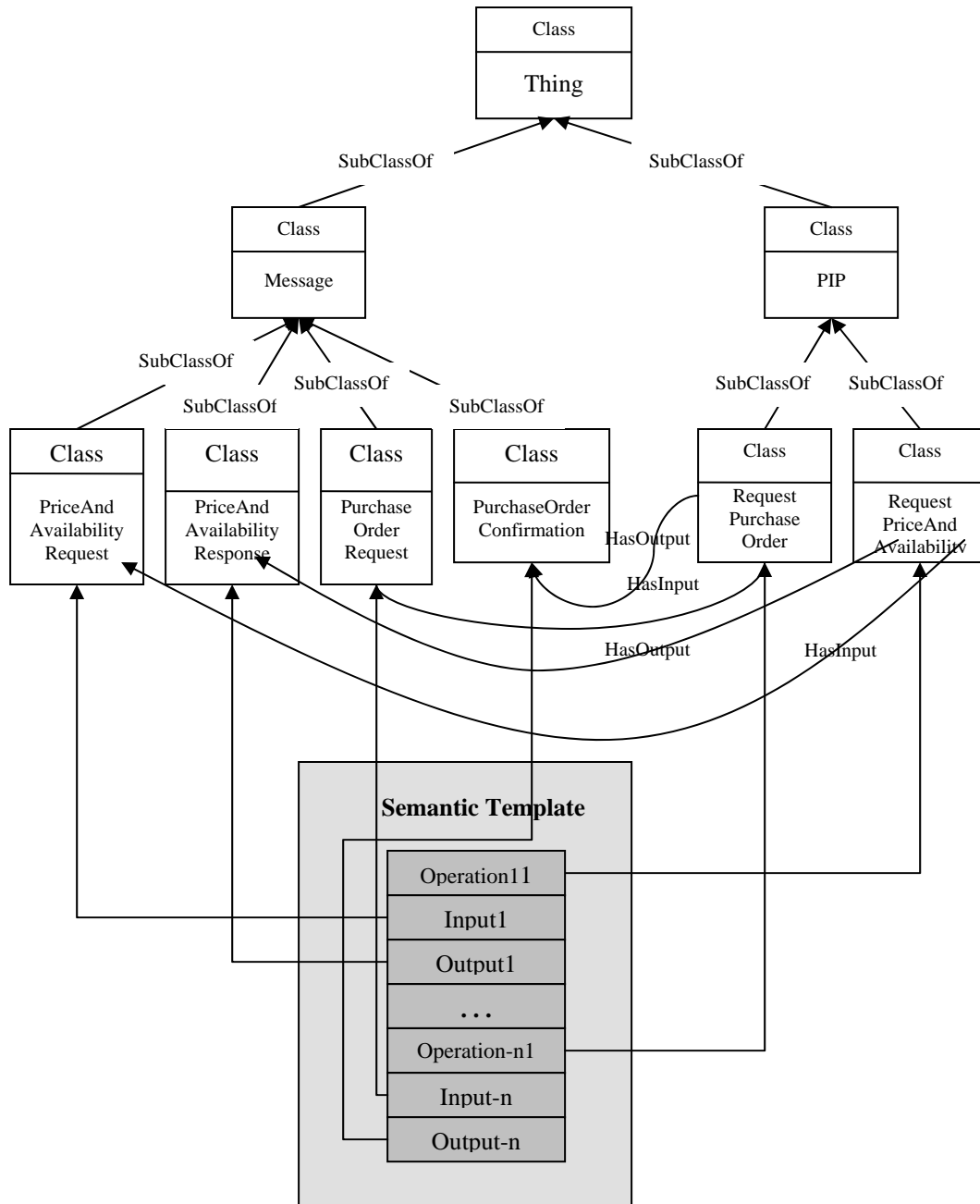
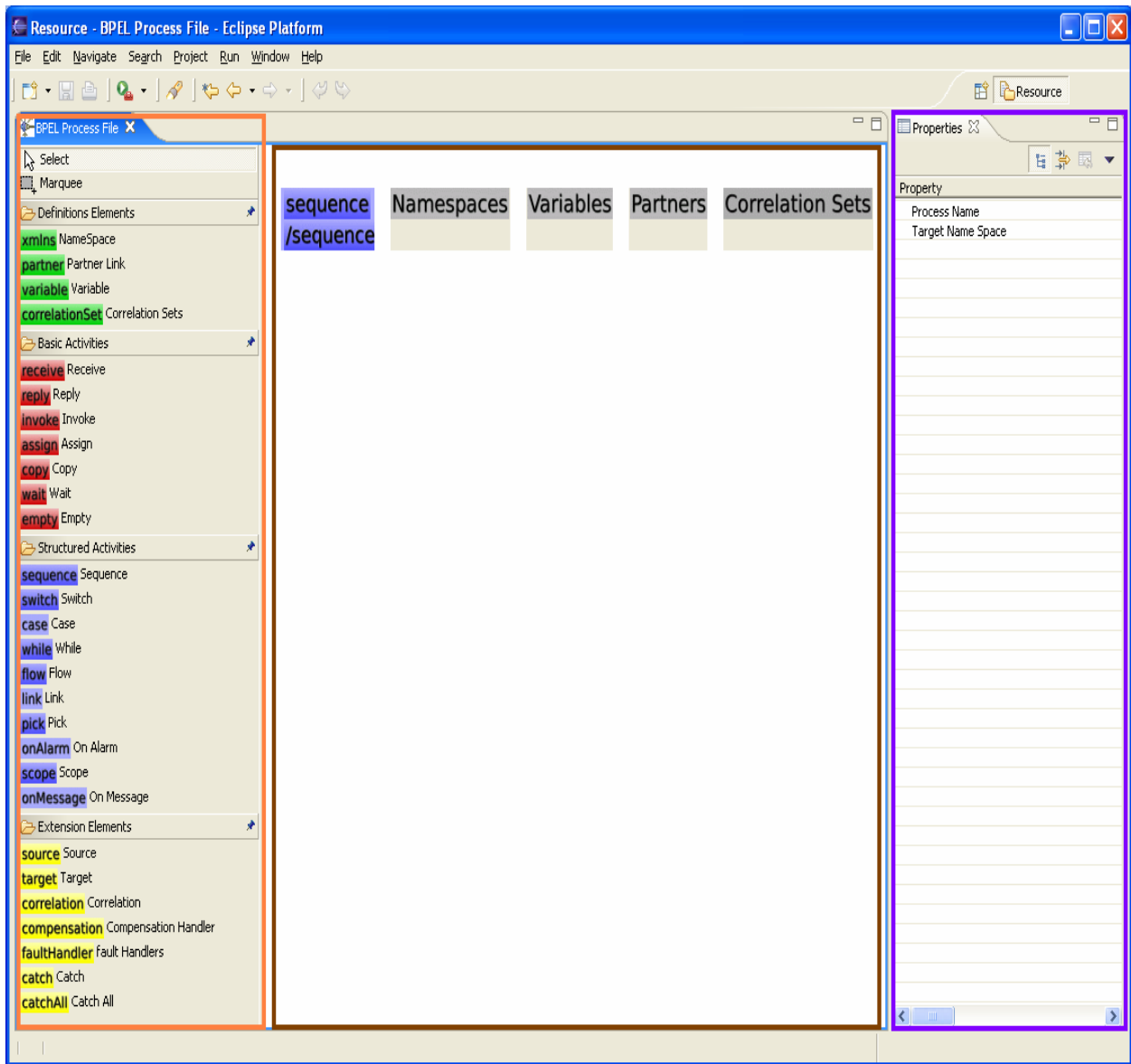


Figure 5: Semantic Template annotated with Ontology Concepts

CHAPTER 3

METEOR-S PROCESS DESIGNER TOOL

This chapter presents the METEOR-S Process Design Tool, which helps a process developer build a BPEL process file using a Graphical User Interface (GUI). The tool offers an easy to use Drag and Drop mechanism for adding process elements to the process being built and then provides a mechanism to edit element properties for elements that are selected by the user for modification. Along with generating a BPEL process file the process also generates a skeleton SDL that can be used for deployment. The process designer needs to slightly modify the process WSDL in case s/he plans to change the process variable types. Currently, all the process variables constitute of a message type with a single string element in them. The Process design and development tool also provides support for dynamic discovery of partners using semantic templates. We discuss the details of each feature in the sections that follow. Figure 6 shows the main user interface of the Process design and development tool.



Color Codes




-  Element Palette
-  Process Canvas
-  Element Property Sheet

Figure 6: METEOR-S Process design and development tool

The Process design and development tool mainly consists of three components described below.

Element Palette

The Element Palette provides the process developer with a palette from which s/he can drag and drop process elements into the process being designed. This palette has entries for all the process elements that are currently supported by the Process design and development tool.

Process Canvas

The Process Canvas forms the main design area for the process. A Process Designer drags and drops process elements from the Element Palette on to the process Canvas. The Canvas offers visual feedback of the current state of the process.

Element Property Sheet

This section of the Process design and development tool offers a means to edit properties of the process element that is currently selected on the Process Canvas. When the Process design and development tool selects a particular element on the canvas, the property sheet gets populated with editable properties of the selected process element.

3.1. MODEL-VIEW-CONTROLLER

The METEOR-S process designer follows the Model View Controller (MVC) [27] architecture. The MVC architecture is a commonly used and effective architecture to build systems having GUI based design. In the MVC paradigm input handling, modeling of the functional logic and the visual feedback of the state of the system are decoupled and handled separately. The “View” handles and manages the graphical output to the UI. The controller interprets the user inputs and acts as a link between the model and the view. The “Model” manages the behavior of the application logic. Figure 7 shows the relationship between Model-View-Controller.

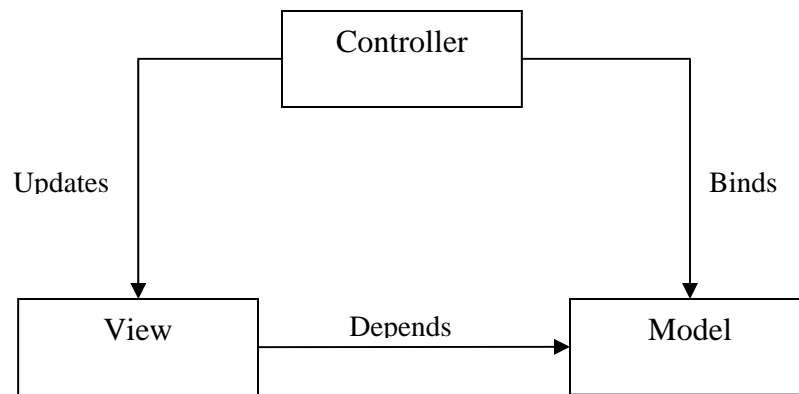


Figure 7: Model-View-Controller Architecture

The Controller interprets mouse and keyboard inputs from the user and translates these user actions into commands that are sent to the model and/or view to reflect the changes in the current state of the system and to give a visual feedback change. The model manages one or more data elements and maintains the state of the system. It responds to queries about its state, and responds to instructions to change state. Typically the business logic of the system resides in the model classes. The view is responsible for presenting data to the user through a combination of graphics and text.

3.2. GRAPHICAL EDITING FRAMEWORK

The Graphical Editing Framework (GEF) which is part of the Eclipse tool integration platform, allows developers to create a rich graphical editor from an existing application model [10]. The METEOR-S Process design and development tool uses the GEF framework to build its graphical user interface. GEF is fully written in Java and hence works on all operating systems which are officially supported by the Eclipse platform [29]. This feature helps in using the METEOR-S Process design and development tool on Eclipse supported platforms without any porting issues. GEF depends on Draw2d which is a lightweight toolkit built with The Standard Widget Toolkit

(SWT) [12] that offers optimized layout and painting along with giving a native look and feel for the GUI. The GEF framework follows the MVC model and hence fits in well with the overall design on METEOR-S Process design and development tool.

3.3. METEOR-S PROCESS DESIGN AND DEVELOPMENT TOOL ARCHITECTURE

The METEOR-S Process design and development tool consists of four main components, UI Layer (View), Controller Layer (Controller), Logic Layer (Model) and the Physical layer (Data Access). The layered structure is shown in figure 8.

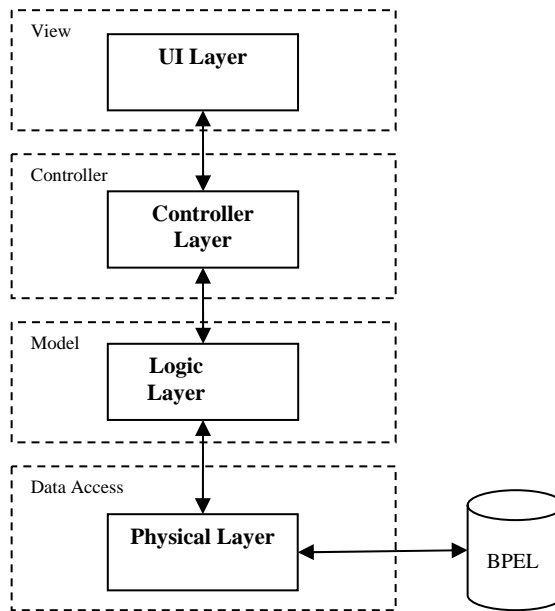


Figure 8: METEOR-S Semantic Process design and development tool Architecture

The UI Layer, as described earlier, takes care of reflecting the current state of the process and that of each contained BPEL element to the user. The Logic Layer represents the in-memory model of the entire process. The UI Layer reflects the current state of the model classes to the

user. The Controller Layer is responsible for handling editing of the BPEL element properties and chaining the Logic and UI layer.

3.3.1. Physical Layer

The Physical layer deals with generating WSBPEL process file and input/output to the secondary storage. When the process model is saved by the user, this module handles the conversion of the in-memory model of the entire process to WSBPEL [2] process file. Similarly, when a WSBPEL process file is requested to be open for editing, the physical layer reads in the process and sets up the in-memory structure for the process. The physical layer components get invoked only during loading an existing process into the designer or saving a process after design completion. The physical layer also generates the process WSDL [13] for the Web process. The WSBPEL engine uses the process WSDL to expose the operations of the process to allow other clients or processes to access the process as a Web service in itself.

3.3.2. Model Layer

The Model Layer forms the core of the designer. The model classes in this layer are responsible for holding the current state of each element of a business process which includes the process as a whole. For each business element or activity defined in the WSBPEL specification a model class has been designed that reflects the structure and properties of the corresponding element as outlined in the WSBPEL specification. The METEOR-S Process design and development tool implements all the process elements proposed in the WSBPEL specification. We have grouped these elements according to their functionality. The following is the grouping which includes a short description of each of the process activities/elements.

Definitions group

This group contains elements that get defined in the process and are later used by other elements of the process. These elements are outlined in table 2.

Table 2: Activities that belong to the Definition group

Element Name	Description
Namespace	Declares a XML namespace that is used to qualify QNames of other process elements.
Variable	These variable objects are operated upon during the process execution. The Variables can either be process variables or messages types of partner services. In order for the variables to be used in other process elements, they should be defined prior in the process.
Partner	These are partner Web Services whose operations are used by the current process.
Correlation Set	Correlation sets form a named group of process properties (ideally variables) that serve to define a way of identifying an application-level conversation within a business protocol instance.

Basic Activities Group

These activities are the most commonly used activities in a business process. They directly interact with the defined data variables, interact with service partners or provide services to be used by other potential partners. These activities are described in table 3.

Table 3: Activities that belong to the Basic activity group

Element Name	Description
Invoke	Used to invoke an operation of a partner Web service.
Receive	The receive activity is an entry point into the process. Typically a pair of receive-reply activities define a single operation exposed by a business process.
Reply	The reply activity acts as successful termination of a process operation.
Wait	The wait activity specifies the process engine to wait either for a specific duration or until a specified time elapses.
Empty	The Empty activity is used for doing nothing. It acts as a no-op operation.
Copy	Used to copy values from one variable to another or from one variable part to another.
Assign	Container for copy activities.

Structured Element

This group mostly constitutes of process elements that have the containment property. These elements contain other process elements as their child elements. The structured activities are listed in table 4.

Table 4: Activities that belong to the structured activity group

Element Name	Description
Sequence	Used for grouping activities that need to be carried out in a sequential order.
Switch	Acts as an exclusive choice or for XOR-split grouping of activities
Case	There are the only allowed child elements of the Switch Activity (conditional branching). They may contain other activities as their children activities.
While	Used for looping through a set of activities depending on specified condition.
Pick	Acts as a simple merge where activities are run in parallel and the first child activity that completes triggers the merge.
Flow	Acts as a Synchronizing Merge.
Link	links are defined inside a flow and are used to connect a source activity to a target activity.
Scope	to define a nested activity in itself.
onMessage, onAlarm	These are the container activities for conditional events inside a pick activity.

Extensional group

This group contains activities that are used to handle exceptions for handling special case conditions. The activities belonging to this group are listed in table 5.

Table 5: Extensional group

Element Name	Description
Source, Target	Define the source and targets of a link activity
Correlation	Used to identify correlation sets that should be used in conjunction with either a receive, reply of a invoke activity
Compensation Handler, Fault Handler	Specify the activities that must be performed in response to faults resulting from the partner service invocations
Catch, CatchAll	Specify particular activity to be carried out depending on the type of fault caught.

Each of the above activity has a corresponding Model class to represent them. These model classes represent the in-memory object for the activities. Figures 9 and 10 show the part corresponding to these model classes of the overall UML diagram for the system.

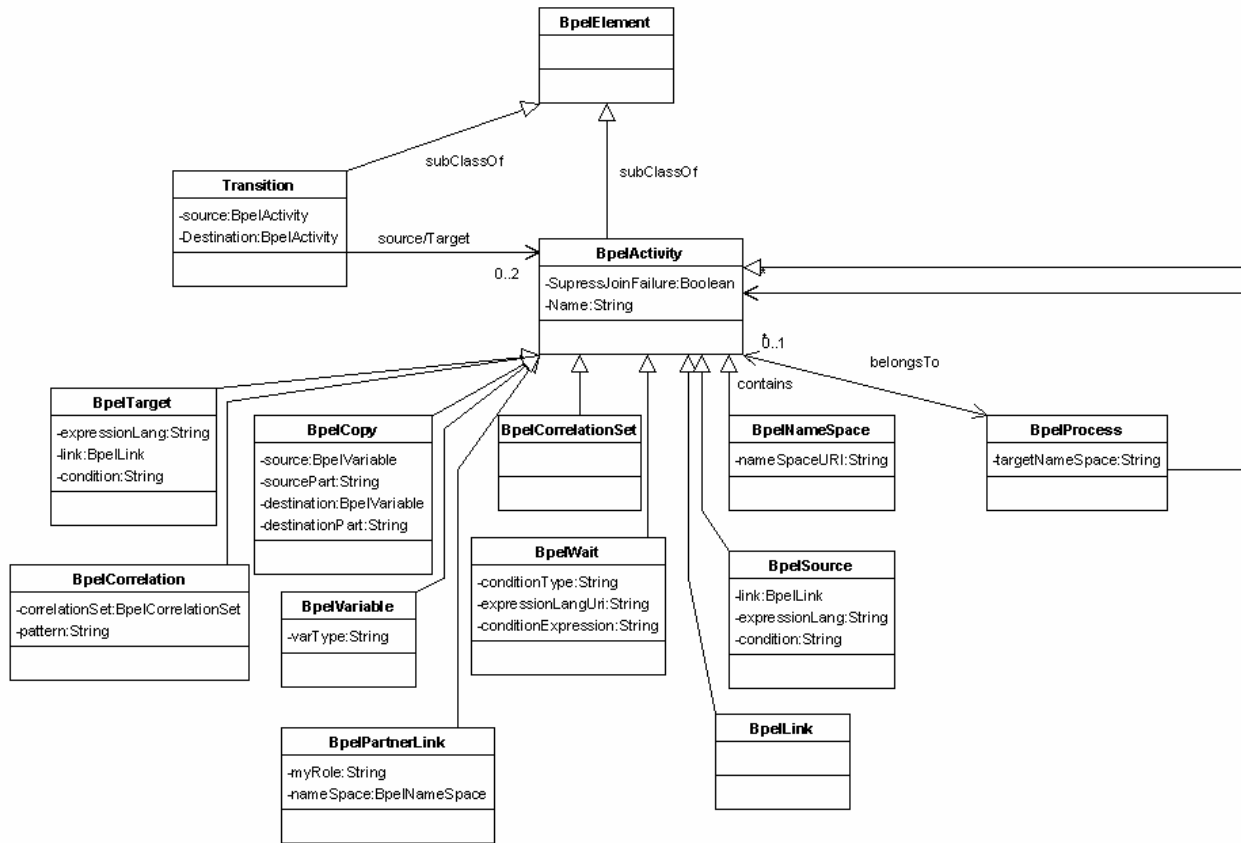


Figure 9: UML diagram for model class, Part I

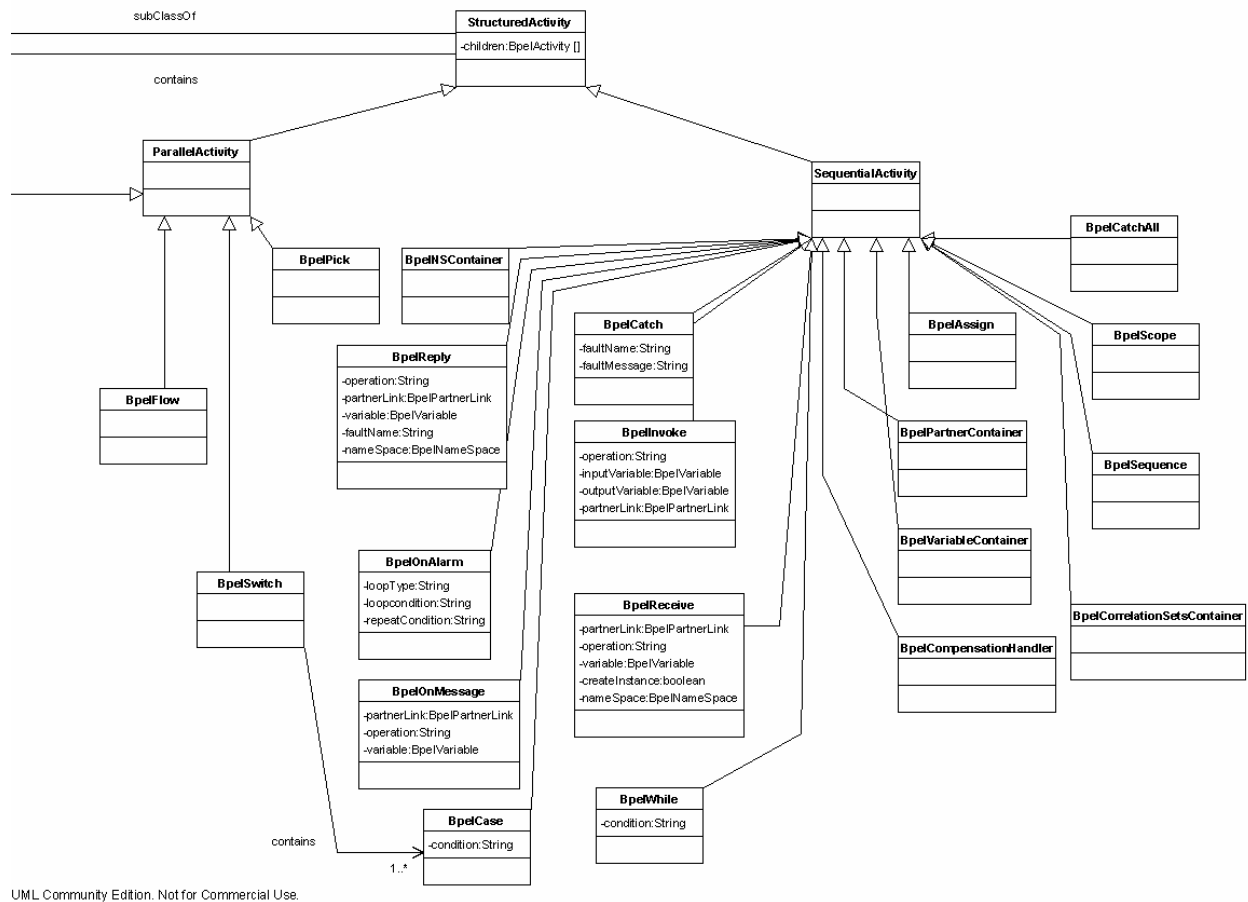


Figure 10: UML diagram for model class, Part II

3.3.3 Controller Layer

A controller class is designed for each model class that needs to be presented in the UI. The controller class, as the name suggests, acts as an agent between the model and the UI presentation. Any changes in the model layer are reflected in the UI with the help of the controller classes. Also, it handles user input in the form of either creation and deletion of new process elements or modifications done to element properties. The control class is also responsible for creating new view objects for each new process element that gets created. The

view class is tied to the corresponding controller and any changes in the model classes get reflected onto the view by the controller class.

3.3.4. UI Layer

The view objects in the UI layer give a visual representation of the process state to the user. The UI layer is responsible for displaying the entire process on the process canvas and placing newly created elements at their right position. As described earlier, the METEOR-S Process design and development tool makes use of GEF to help with UI layout. Since GEF takes care of UI related activities such as action listeners, dragging and dropping it helps in eliminating the effort of writing Java code of classes to handle such activities.

3.4. INFRASTRUCTURE

The previous sections presented the features and functionality of the METEOR-S Process design and development tool. In this section, we discuss the infrastructure, i.e., the various libraries and packages used by the tool. Table 6 gives a different packages/libraries used while developing the designer tool.

Table 6: Packages and tools used

Package/Tool	Usage
Eclipse	The main development platform on which the tool is deployed
GEF	Framework used for rendering process elements
BPWS4J	Library to generate WSBPEL Process file and parse an existing file
WSDL4J	Used for generation process WSDL for deployment of the process.
UDDI4J	Registry to dynamic discovery of partner services
JWSDP4J	UDDI Registry for discovering semantic Web services

3.5. IMPLEMENTATION

The Data Access layer of the METEOR-S Process design and development tool uses a WSBPEL implementation API to generate and parse a WSBPEL process file. Currently, the WSBPEL Parser/Generator module uses the BPWS4J API as its base library. The design allows the flexibility to replace the API library with a different implementation. Currently the tool uses the BPWS4J [40] API. Each model class in the model layer is additionally backed up by a corresponding class from the WSBPEL API used in the Data Access Layer module.

CHAPTER 4

FEATURES OF THE METEOR-S PROCESS DESIGN TOOL

This chapter discusses some of the salient features of the process design and development tool. Apart from offering usability features of a user-friendly GUI, the tool also makes use of efforts in the area of Semantic Web Services by providing an infrastructure for semantic Web service discovery and intergration.

4.1. SEMANTIC SELECTION OF PROCESS PARTNERS

To support capabilities for dynamic partner selections, the METEOR-S Process design and development tool makes use of Semantic Web Services. For the semantic selection process to work, it is assumed that semantically annotated services have been developed and published by Web Service providers. The Web Service designer annotates the data and operations of a Web Service before publishing them to a UDDI registry. METEOR-S uses OWL ontologies to annotate the operations offered by a Web service and the parameters and return types of those operations. This can be done using the WSDL-S [21] development module of METEOR-S and is discussed in more detail in [7].

The selection process can be thought of being constituted of two phases. First is the template generation phase where the process developer specifies his requirements for a partner in terms of semantic template. In the second phase, the METEOR-S Process design and development tool tries to find a matching partner for the semantic template specified. The following sections discuss the two phases in detail.

In the first phase, the process developer generates a semantic template with a simple to use GUI. The developer can add operations to the template and specify the input and output messages for each operation. Each of these entities is annotated by ontology concepts. The developer can either create a new template or s/he has a choice to load an existing template. The Process design and development tool currently offers selection of a pre-designed semantic template to be selected for partner discovery. However, as part of future work, a suitable user interface has been proposed to be an integrated part of the Process design and development tool.

Phase two utilizes the core searching mechanism for dynamic partner discovery. Once the Process design and development tool has generated/selected a semantic template for a particular partner, s/he can trigger the discovery process to find a matching partner. The METEOR-S Process design and development tool extracts the semantic information from the template and passes it to the discovery module. The discovery module performs the semantic search and returns results that match the template. The process developer can then choose a particular service from the result set to act as a partner for the particular process instance. This phase is explained in figure 11. The process developer can either discover the partner service at design time or defer it till deployment time. Provision to allow runtime discovery is discussed as part of future work.

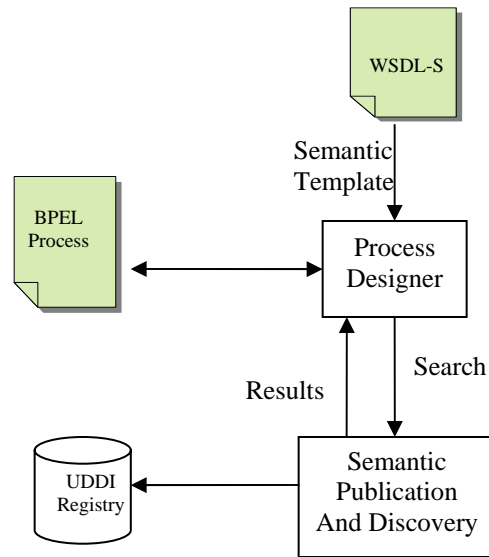


Figure 11: Phase II – Dynamic partner discovery

4.1.1. Design Time Discovery vs. Deployment Time Discovery

As discussed earlier, a process developer can either choose to discovery partner services at design time or may do so at deployment time of the process. The advantages of deployment time discovery is that the level of dynamism achieved is greater as compared to design time discovery. Chances are that between the interval of design an execution, some services that yield a better match may be available. However, with deployment time discovery, one may have to look into the issue of data mapping of the inputs and outputs of the new found partner with that of what the process expects. This is discussed in further details in [11].




4.2. USABILITY FEATURES

The METEOR-S Process design and development tool offers an easy to use GUI for process developers to rapidly build Web processes. Processes developers are offered support for dragging and dropping of process elements on a process “canvas”. This, combined with ease of element selection and deletion, offers a simple to use GUI. Selecting a particular element opens up a property sheet that allows the user to modify element properties. This approach helps in hiding the unnecessary syntactic details from the developer.

4.2.1. Color coded process activities

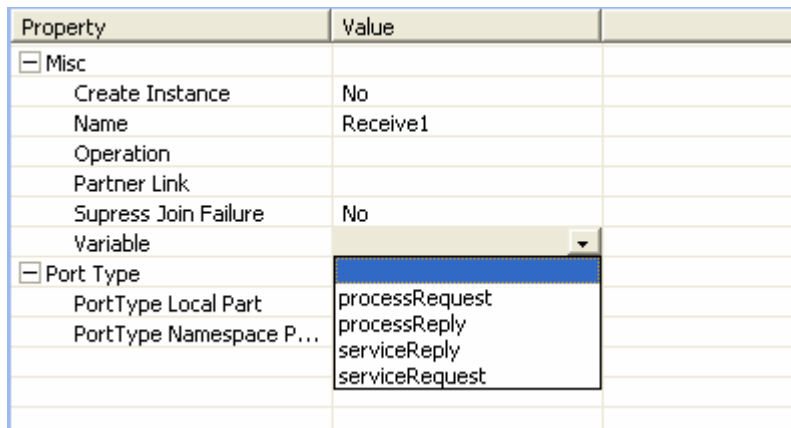
As discussed earlier, the process activities in the Process design and development tool have been categorized by functionality. To help a process developer differentiate the activities easily, each category of process element has been given a distinct color. This color coding helps in easy identification of process elements on the process canvas and makes process design more intuitive. Table 7 lists the colors used for the different categories.

Table 7: Color codes for activity groups

Group	Color
Basic Activities	RED 
Structured Activities	BLUE 
Extensional Activities	YELLOW 
Definitions	GREEN 

4.2.2. Definition lookup

Many of the basic activities of the business process have properties that refer to definition elements (Variables, Partnerlinks or XML namespaces). To avoid errors that may arise due to mistakes in entering names of these elements, the Process design and development tool offers a drop-down box of available choices for the entries. For example, activities that have a variable as one of its properties would be offered a drop-down box of all the variables that have been defined for the process so far. This also helps in ensuring correctness of the system, since a process developer will not be able to set definition elements for an activity property unless they are first added to the process. Figure 12 provides a sample illustration of this feature.



Property	Value	
[-] Misc		
Create Instance	No	
Name	Receive1	
Operation		
Partner Link		
Supress Join Failure	No	
Variable		
[-] Port Type		
PortType Local Part	processRequest	
PortType Namespace P...	processReply	
	serviceReply	
	serviceRequest	

Figure 12: Definition lookup

4.2.3. Avoiding ambiguous process designs

Before allowing addition of a new elements to a container type element, The METEOR-S Process design and development tool checks to see if the insertion is safe. This makes sure that the process does not end up in an ambiguous state due to invalid activities. For example, the

only valid addition to an activity of type switch is a case activity. An attempt to insert any other activity as a child of a switch activity is prohibited.

4.2.4. Designing complex processes

Container elements such as sequence, flow, etc. can be nested to any depth. This helps in generating a process with arbitrary complexity. Further, use of view objects with borders and distinct begin and end tags for each container elements makes it easy to trace the control flow even for complex processes.

4.2.5. Intuitive help messages

Though the METEOR-S Process design and development tool abstracts the WSBPEL syntax from the process developer, chances are that the developer may find it hard to know the meaning/purpose of activity properties. The METEOR-S Process design and development tool provides the developer with descriptive messages for many of the editable properties of process elements. Whenever the process developer clicks on a particular property to be edited, a help message (if set for the particular property) appears in the status bar of the eclipse IDE. Figure 13 shows an illustration of such a help message.

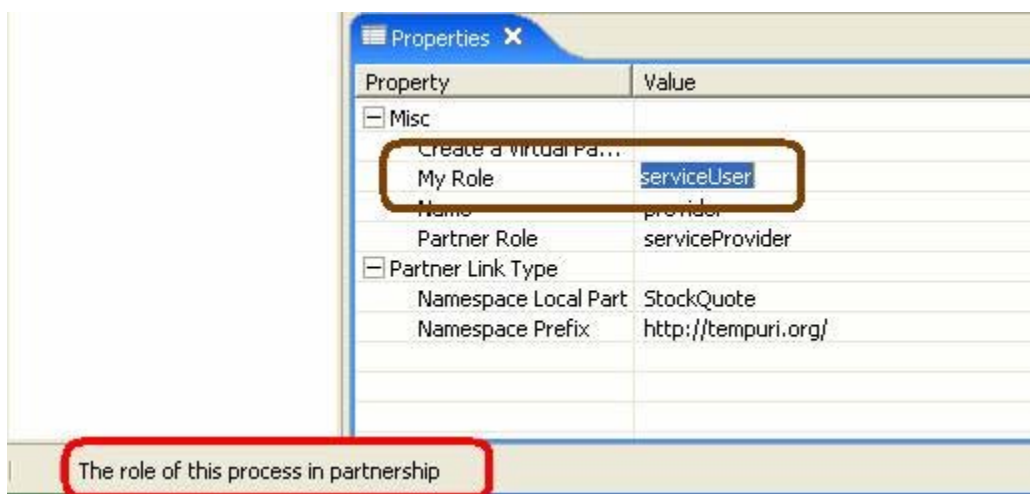


Figure 13: Status bar helper messages

CHAPTER 5

RELATED WORK

In this thesis, we have presented an approach to model business processes using an easy to use Process design and development tool. We also introduced an approach for dynamic partner search using Semantic templates based upon the Semantic Web research. This section discusses some of the related work in the field.

Searching for Web Services using non-semantics approaches like similarity searches, have been proposed [17] and have been evaluated to do better than conventional keyword search. Use of semantics in achieving automation of Web processes creation and Web Service interactions has been proposed [15, 9, 5] using ontologies to describe Web Service entities. The work in [15, 9, 5] follows fully automated composition approach. In these approaches, the process developer has no control over the exact steps of the Web process. The METEOR-S Process design and development tool uses a semi-automatic composition approach that requires human intervention. Though the amount of work needed on the part of the developer in case of fully automated process design is less, current businesses may not be comfortable with the approach. Businesses would expect to have a control over the entire process design. In the METEOR-S Process design and development tool, the process designer has freedom to write the complete process as per his design and we provide dynamism at a finer granularity of partner selection. Though this involves more work on the part of the process developer, s/he has the complete control over the entire process design.

In the area of Process design and development tool without dynamic selection, different approaches to model of Web processes have been proposed [14, 6] either using existing methodologies like UML activity graph [18] or similar workflow languages. Also, some commercial products [30, 31] offer a process design and development tool as part of their business process solutions product.

Table 8 below gives a comparison the different approaches used for modeling Web processes. Table 9 compares features offered by the METEOR-S process design and development tool with other commercially available products. Support for a particular feature is marked by a '+' symbol and the '-' symbol points lack of support for the feature. A partially supported feature is marked by '+/-'.

Table 8: Comparison of Web process modeling approaches

Feature	METEOR-S PDDT	Automated Composition (SWORD, OWL-S)	Commercial Tools (WebSphere, Collaxa)
Usability	+	+/-	+
Design Freedom	+	-	+
API Independence	+	+	-
Target Dependency	+/-	+	-
License	Open Source	?	Commercial

Table 9: Comparison of BPEL designing Tools

Feature	METEOR-S PDDT	WebSphere, Oracle
Service Binding	Support for Late/Dynamic binding	Static Binding
Use of Semantics	Yes	No

CHAPTER 5

USE CASE

In this chapter, we describe a use case and an end to end scenario illustrating the use METEOR-S Process design and development tool. Consider a scenario where the inventory department of an organization plans to write a Web process to order parts for the inventory. The steps involved in such a purchase order process are:

- Get quantity of items to be ordered
- Contact the supplier service and place an order of the desired quantity
- Simultaneously contact the Web Service of a shipping company requesting the shipment of the order items
- The cost of the order items and shipment charges are then supplied to another service to find the total cost of the purchase order.

Figure 14 represents a flow chart of steps to be taken for inventory ordering.

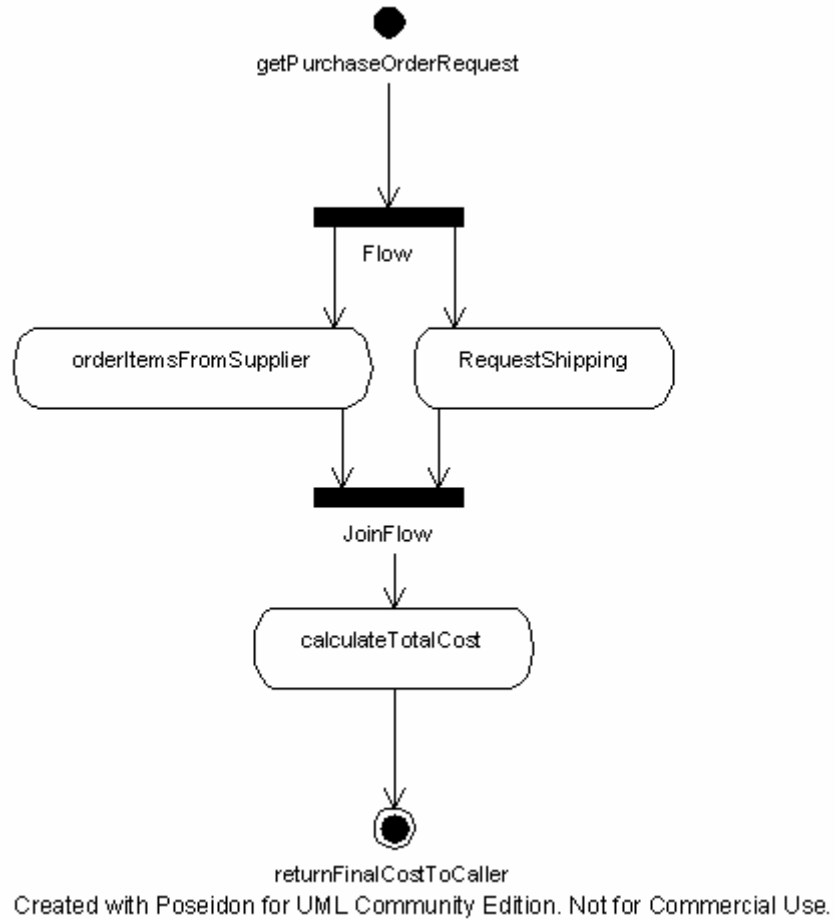


Figure 14: Steps for purchase order process

Such a business process can be designed using the METEOR-S Process design and development tool. A screenshot showing the various elements added to realize the process is shown in figure 15. Once the process is design, the tool saves the process to a BPEL file and generates a process WSDL for the process.

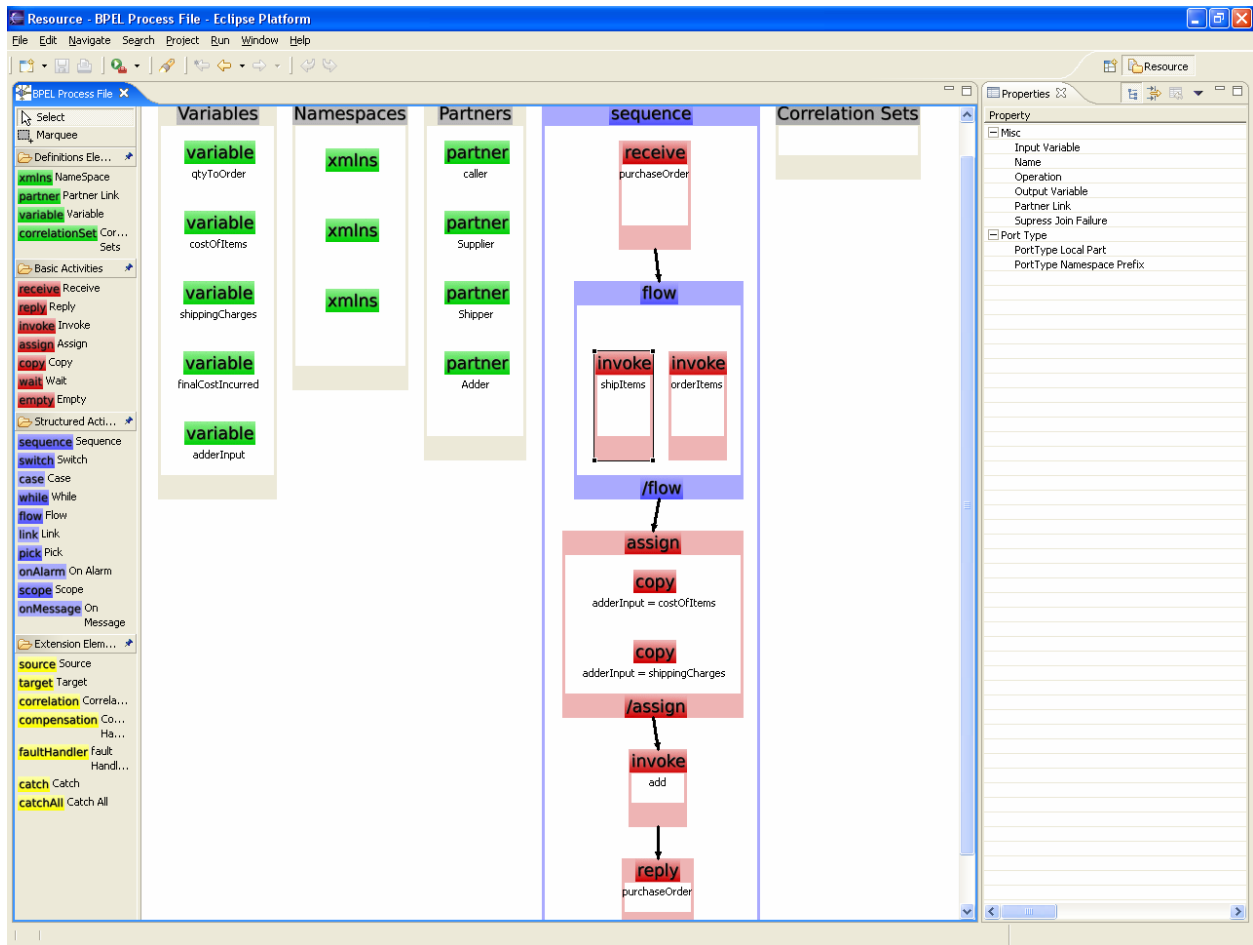


Figure 15: Use Case – Purchase Order Process

Figure 16 and 17 show listings of the generated BPEL file and the generated process WSDL respectively.

```
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  name="purchaseOrder"
  targetNamespace="urn:purchaseOrder"
  xmlns:tns="urn:purchaseOrder">
  <partnerLinks>
    <partnerLink name="caller" partnerLinkType="tns:purchaseOrderPT"
myRole="serviceUser" partnerRole="serviceProvider"/>
    <partnerLink name="Supplier"
xmlns:ns1="http://128.192.251.228:8080/axis/OrderItems.jws"
partnerLinkType="ns1:OrderItems" myRole="buyer" partnerRole="seller"/>
    <partnerLink name="Shipper"
xmlns:ns2="http://128.192.251.228:8080/axis/Shipper.jws"
partnerLinkType="ns2:Shipper" myRole="addressee" partnerRole="mailMan"/>
    <partnerLink name="Adder"
xmlns:ns3="http://128.192.251.228:8080/axis/Adder.jws?wsdl"
```

```

partnerLinkType="ns3:Adder" myRole="additionRequester"
partnerRole="mathematician"/>
</partnerLinks>
<variables>
  <variable name="qtyToOrder"
xmlns:ns4="http://128.192.251.228:8080/axis/OrderItems.jws"
messageType="ns4:orderItemsRequest"/>
  <variable name="costOfItems"
xmlns:ns5="http://128.192.251.228:8080/axis/OrderItems.jws"
messageType="ns5:orderItemsResponse"/>
  <variable name="shippingCharges"
xmlns:ns6="http://128.192.251.228:8080/axis/Shipper.jws"
messageType="ns6:shipItemsResponse"/>
  <variable name="finalCostIncurred"
xmlns:ns7="http://128.192.251.228:8080/axis/Adder.jws?wsdl"
messageType="ns7:addResponse"/>
  <variable name="adderInput"
xmlns:ns8="http://128.192.251.228:8080/axis/Adder.jws?wsdl"
messageType="ns8:addRequest"/>
  <variable/>
</variables>
<correlationSets>
</correlationSets>

<sequence>
  <receive name="initiatePurchaseOrder"
    partnerLink="caller" portType="tns:purchaseOrderPT"
operation="purchaseOrder"
    variable="qtyToOrder" createInstance="yes">
  </receive>
  <flow>
    <invoke name="requestItems"
      partnerLink="Supplier"
xmlns:ns9="http://128.192.251.228:8080/axis/OrderItems.jws"
portType="ns9:OrderItems" operation="orderItems"
      inputVariable="qtyToOrder" outputVariable="costOfItems">
    </invoke>
    <invoke name="requestShipping"
      partnerLink="Shipper"
xmlns:ns10="http://128.192.251.228:8080/axis/Shipper.jws"
portType="ns10:Shipper" operation="shipItems"
      inputVariable="qtyToOrder" outputVariable="shippingCharges">
    </invoke>
  </flow>
  <assign >
    <copy>
      <from variable="costOfItems" part="orderItemsReturn"/>
      <to variable="adderInput" part="num1"/>
    </copy>
    <copy>
      <from variable="shippingCharges" part="shipItemsReturn"/>
      <to variable="adderInput" part="num2"/>
    </copy>
  </assign>
  <invoke name="math"
    partnerLink="Adder"
xmlns:ns11="http://128.192.251.228:8080/axis/Adder.jws?wsdl"

```

```

portType="ns1:Adder" operation="add"
    inputVariable="adderInput" outputVariable="finalCostIncurred">
    </invoke>
    <reply partnerLink="caller" portType="tns:purchaseOrderPT"
operation="purchaseOrder"
        variable="finalCostIncurred">
    </reply>
</sequence>
</process>

```

Figure 16: Use Case – Purchase Order BPEL

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:purchaseOrder"
    xmlns:tns="urn:purchaseOrder"
    xmlns:ns1="http://128.192.251.228:8080/axis/OrderItems.jws"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:ns2="http://128.192.251.228:8080/axis/Adder.jws?wsdl"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

    <wsdl:portType name="purchaseOrderPT">
        <wsdl:operation name="purchaseOrder">
            <wsdl:input message="ns1:orderItemsRequest"/>
            <wsdl:output message="ns2:addResponse"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:service name="purchaseOrderBP">
</wsdl:service>

        <plnk:partnerLinkType name="purchaseOrderPT">
            <plnk:role name="serviceUser">
                <plnk:portType name="purchaseOrderPT"/>
            </plnk:role>
        </plnk:partnerLinkType>
        <plnk:partnerLinkType name="OrderItems">
            <plnk:role name="buyer">
                <plnk:portType name="purchaseOrderPT"/>
            </plnk:role>
        </plnk:partnerLinkType>
        <plnk:partnerLinkType name="Shipper">
            <plnk:role name="addressee">
                <plnk:portType name="purchaseOrderPT"/>
            </plnk:role>
        </plnk:partnerLinkType>

```

```
<plnk:partnerLinkType name="Adder">
  <plnk:role name="additionRequester">
    <plnk:portType name="purchaseOrderPT"/>
  </plnk:role>
</plnk:partnerLinkType>
</wsdl:definitions>
```

Figure 17: Use Case – Purchase Order Process WSDL

Using the above generated BPEL process file and the process WSDL, the process can be deployed using a BPEL engine. The process can now be invoked as a regular Web Service using the process WSDL exposed by the engine.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this work, we presented the current research interests and developments in the area of Web processes. We discuss the importance of incorporating Web Services in business solutions to achieve better integration within organizational units of a company or across organizational boundaries. We stressed on the issue that with demand for design of complex business processes, there is an increasing interest in tools that support designing such Web Service based processes. This work provides original contribution in the area of business process design in the form the METEOR-S Process design and development tool developed as an Eclipse plugin. The tool offers a GUI based approach to model business process in the form of a BPEL process. This work highlights some key usability features of the tool that assist process developers in designing complex business processes without getting into syntactical details of BPEL. The tool has user-friendly features that help prevent common design mistakes and thus help the process designer in focusing on the process flow rather than getting confused with minor details. The advantages of incorporating semantics in the process of partner discovery led to the discussion of having a facility for dynamic partner selection in a Process design and development tool. This work uses a semantic template based approach to achieve dynamic partner discovery, thus, allowing process developers to optimize their processes by facilitating partner search at design time or deployment time.

The primary contribution of this work is the development of a Graphical User Interface based tool to design WSBPEL business process to model Web processes. The METEOR-S Process design and development tool offers ease of development, while hiding the syntactic

details from the process developer. The designer gives complete freedom to the process developer over the design and structure of the Web process s/he plans to design.

6.1. FUTURE WORK

The METEOR-S Process design and development tool can be further improved to make it more user-friendly and intuitive for designing business processes. If a process designer plans to use static partners instead of specifying a semantic template, a UDDI registry browser could be provided to browse a UDDI service registry to search for partners. Also, for use of dynamic discovery, support for integrating a semantic template generation module is proposed. This would help the process developer either select a pre-existing semantic template or generate a new one from within the Process design and development tool.

As part of the future work, a process developer can be provided with elements of a process partner's WSDL at desired places. For example, currently to add a new variable to the process, the process developer is expected to type in the name of the message s/he wishes to use for the partner WSDL. Instead, a module to parse the partner WSDL on the fly and provide a drop-down menu of available messages in the partner WSDL can be added to further avoid errors due to typographical mistakes.

While using the feature of dynamic partner discovery, especially at deployment time, it is expected that the partners have the same data bindings as the virtual service assumed. Adding dynamic data mapping capabilities to further enhance deployment time binding is proposed, which would allow developers to incorporate service partners that do not exactly subscribe to the assumed data binding. The Process design and development tool can be adapted to further achieve run-time discovery of partner services. Such dynamic data binding capabilities also open up the option of delaying the discovery phase to process run time instead of design or

deployment time. This can be done by use of proxy services that perform the discovery and invocation of partner services on behalf of the process. Currently, the WSBPEL engines available do not provide capabilities to identify the current state of a running process, i.e., provide real time monitoring capabilities to track of activities that have been completed, ones that are in progress and the ones that have still not been started. It has been proposed to find ways to hook the Process design and development tool into such process engines so as to offer visual feedback of the current status of the process for monitoring purposes.

REFERENCES

- [1] Universal Description, Discovery and Integration (UDDI), <http://www.uddi.org/>
- [2] Web Services Business Process Execution Language (WSBPEL), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- [3] METEOR-S: Semantic Web Services and Processes, <http://lstdis.cs.uga.edu/projects/METEOR-S/>
- [4] Narayanan S., and McIlraith S., "Simulation, Verification and Automated Composition of Web Services" Proceedings of the eleventh international conference on World Wide Web, May 2002, Pages: 77 - 88
- [5] Traverso P., and Pistore M., "Automated Composition of Semantic Web Services into Executable Processes", Proceedings of the 3rd International Semantic Web Conference (ISWC2004), pp. 380-394
- [6] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. QUT Technical report, FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002
- [7] Rajasekaran P., Miller J., Verma K., Sheth A., "Enhancing Web Services Description and Discovery to Facilitate Composition", Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition, July 2004, pp. 55-68
- [8] J. Miller, D. Palaniswami, A. Sheth, K. Kochut, H. Singh, "WebWork: METEOR's Web-based Workflow Management System", Journal of Intelligence Information Management Systems, 1997, pp. 185-215

- [9] Nau D., Cao Y., Lotem A., and Muñoz-Avila H, "SHOP: Simple Hierarchical Ordered Planner", Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999, Pages 968 - 975
- [10] GEF: Graphical Editing Framework, <http://www.eclipse.org/gef/>
- [11] Verma K., Akkiraju R., Goodwin R., Doshi P., and Lee J., "On Accommodating Inter Service Dependencies in Web Process Flow Composition", 2004 AAAI Spring Symposium Series, March 2004, pp 37-43
- [12] Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001, pp. 34-43
- [13] WSDL: Web Service Description Language, <http://www.w3.org/TR/wsdl>
- [14] Zarras A., Vassiliadis P., and Issarny V., "Model-Driven Dependability Analysis of Web Services", International Symposium on Distributed Objects and Applications, October 2004, pp. 69-79
- [15] Ponnekanti S., and Fox A., "SWORD: A Developer Toolkit for Web Service Composition", In Eleventh World Wide Web Conference (WWW2002, Web Engineering Track), Honolulu, Hawaii, May 2002.
- [16] Dogac A., "Exploiting Semantic of Web Services through ebXML Registries", Keynote Talk, 14th International Workshop on Research Issues on Data Engineering, Boston, 2004, http://www.srdc.metu.edu.tr/~asuman/Dogac_RIDE_04_KeynoteAddress.ppt
- [17] Dong X., Halevy A., Madhavan J., Nemes E., and Zhang J., "Similarity Search for Web Services", 30th VLDB Conference, August - September 2004, pp. 372-383
- [18] UML: Unified Modeling Language, Object Management Group, <http://www.uml.org/>

- [19] Paolucci M., Sycara K., and Kawamura T., "Delivering Semantic Web Services" In Proceedings WWW2003, May 2003, pp. 829
- [20] Sivashanmugam K., Verma K., Sheth A., and Miller J., "Adding Semantics to Web Services Standards", 1st International Conference on Web Services, June 2003, pp. 395-401.
- [21] Miller J., Verma K., Rajasekaran P., Sheth A., Aggrawal R., and Sivashanmugam K., "WSDL-S: A Proposal to W3C WSDL 2.0 Committee", <http://lsdis.cs.uga.edu/projects/WSDL-S/wSDL-s.pdf>
- [22] Aggarwal R., Verma K., Miller J., and Milnor W., "Constraint Driven Web Service Composition in METEOR-S" Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004), Shanghai, China (September 2004) pp. 23-32.
- [23] Verma K., Sivashanmugam K., Sheth A., Patil A., Oundhakar S., and Miller J., "METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services", Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers
- [24] Aggarwal R., Verma K., Miller J., and Milnor W., "Dynamic Web Service Composition in METEOR-S", Technical Report, LSDIS Lab, Computer Science Dept., UGA, May 2004.
- [25] Sirin E., Parsia B., and Hendler J., "Composition-driven filtering and selection of semantic Web services", AAAI Spring Symposium on Semantic Web Services, 2004, pp. 129-138
- [26] Sivashanmugam K., Miller J., Sheth A., and Verma K., "Framework for Semantic Web Process Composition", International Journal of Electronic Commerce (IJEC), Special Issue on Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce, Vol. 9, No. 2 (Winter 2004-5) pp. 71-106. M.E. Sharpe, Inc.

- [27] Reenskaug T., "The Model-View-Controller (MVC) Its Past and Present", http://heim.ifi.uio.no/~trygver/2003/javazone-jao0/MVC_pattern.pdf
- [28] OWL-S: Semantic Markup for Web Services, <http://www.daml.org/services/owl-s/1.0/owl-s.html>
- [29] Eclipse: Open platform for tool integration, <http://www.eclipse.org/>
- [30] Oracle BPEL Process Manager, <http://www.oracle.com/technology/products/ias/bpel/index.html>
- [31] IBM WebSphere: <http://www-306.ibm.com/software/websphere/>
- [32] A. Sheth, "Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration", Invited Talk WWW 2003 Workshop on E-Services and the Semantic Web, Budapest, Hungary, May 2003, http://www.ics.forth.gr/isl/essw2003/talks/seth_essw_semanticwebprocess.htm
- [33] K. Sivashanmugam, A. Sheth, J. Miller, K. Verma, R. Aggarwal, P. Rajasekaran, "Metadata and Semantics for Web Services and Processes", 2003, Book Chapter, Datenbanken und Informationssysteme: Festschrift zum 60- Geburtstag von Gunter Schlageter, Benn et al Eds, Praktische Informatik I, Hagen, pp. 245-272.
- [34] A. Patil, S. Oundhakar, A. Sheth and K. Verma, "METEOR-S Web service Annotation Framework", World Wide Conference, In the Proceedings of the 13th W3C Confernece, New York, USA, 2004, pp. 553-563.
- [35] K. Sivashanmugam, K. Verma and A. Sheth, "Discovery of Web Services in a Federated Registry Environment", 2004, Proceedings of IEEE Second International Conference on Web Services, San Diego, California, USA, pp. 270-278.

- [36] Dumas M., and ter Hofstede A. H. M, “UML Activity Diagrams as a Workow Speci_cation Language”, *Lecture Notes in Computer Science*, vol. 2185, pp. 76–90, 2001
- [37] WS-Policy: Web Services Policy Framework,
http://www-128.ibm.com/developerworks/library/specification/ws-polfram/
- [38] SWT: The Standard Widget Toolkit, *http://www.eclipse.org/swt/*
- [40] BPWS4J: The IBM Business Process Execution Language for Web Services Java™ Run Time, *http://www.alphaworks.ibm.com/tech/bpws4j*
- [41] ActiveBPEL, Open Source BPEL Server, *http://www.activebpel.org/*
- [42] Kochut K., Sheth A., and Miller J., “ORBWork: A COBRA-Based Fully Distributed Scalable and Dynamic Workflow Enactment Service for METEOR”, Technical Report #UGA-CS-TR-98-006, Department of Computer Science, University of Georgia, 1998
- [43] METEOR: Managing End-To-End OpeRations,
http://lstdis.cs.uga.edu/Projects/past/METEOR/
- [44] Simple Object Access Protocol (SOAP) 1.1, *http://www.w3.org/TR/soap/*
- [45] Peltz C, “Web Services Orchestration and Choreography”, *Web Services Journal*, Volume 03 Issue 07, July 2003, pages 30-35
- [46] Doshi P., Goodwin R., Akkiraju R., and Verma K., “Dynamic Workflow Composition using Markov Decision Processes”, *International Journal of Web Services Research*, 2005, pp. 1-17
- [47] RosettaNet: *http://www.rosettanet.org*
- [48] Azami M., RosettaNet Ontology, *http://lstdis.cs.uga.edu/~azami/pips.html*
- [49] Kitamura Y., and Mizoguchi R., "Functional Ontology for Functional Understanding", Twelfth International Workshop on Qualitative Reasoning (QR-98), Cape Cod, USA, AAAI Press, 1998, pp.77-87

[50] Gardner D., Knuth K.H., Abato M., Erde S.M., White T., DeBellis R., and Gardner, Common data model for neuroscience data and data model interchange. J. Am. Med. Informatics Assoc. 8(1): 17-33, 2001

APPENDIX A: INSTALLATION GUIDE

REQUIREMENTS

1. JDK 1.4, <http://java.sun.com/j2se/1.4.2/>

NOTE: This tool cannot work with JDK 5 due to issues with third party libraries.

To configure Eclipse to JDK 1.4 use the following instructions

- a. Open Eclipse and make sure you are in the Resource perspective (window -> open perspective -> others -> Resource)
- b. Check if JDK 1.4 has been recognized as installed JRE (window -> preferences -> java -> installed JREs). If you don't find JDK 1.4 in there add a new JRE pointing to your JDK 1.4 installation.
- c. set the JDK 1.4 JRE as default

2. Eclipse 3.0 or later, <http://www.eclipse.org/>

3. GEF 3.0.1 plugin for Eclipse, <http://www.eclipse.org/gef/>

Follow the instructions on eclipse homepage to install into GEF

INSTALLATION

1. Download the BpelDesigner Project from here.

2. There is a dependencies plugin which has all the third party libraries needed by the METEOR-S Process Designer Tool.

The plugin is shipped in the zip archive, extract the plugin (directory named, dependencies_1.0.0) and copy it into the Eclipse's plugin Folder.

NOTE: Due to some technical issues, currently the Process Designer Tool cannot be deployed as plugin. We plan to fix this issue in our next release.

3. Extract the "Bpel Designer" project directory from the zip archive and copy it into the Eclipse workspace directory.

4. Start Eclipse and import the bpelDesigner project into workspace.

This can be done as follows:

File -> Import -> Existing project into workspace

On the project import screen, browse to the Eclipse workspace folder and select "Bpel Designer" directory. Click Finish

5. The project is now imported into the workspace.

6. Get into the Java Perspective (window -> open perspective -> others -> Java)

7. Compile the project and run as a new configuration. To run the project:

Run -> Run... -> Select "Run time workbench" and click on new and then click run. (Eclipse 3.0)

Run -> Run... -> Select "Eclipse Application" and click on new and then click run. (Eclipse 3.1)

(This is needed only for the first run of the project, in subsequent runs, the configuration would be remembered by Eclipse).

6. Now the METEOR-S Process design and development tool is ready to be used. Please refer to the User's Guide for using the tool.

APPENDIX B: USERS GUIDE

STARTING ECLIPSE

Go to the eclipse directory and run the eclipse executable

STARTING METEOR-S BPEL DESIGNER

Follow in instruction in the installation guide to load the bpelDesigner project.

Run the bpelDesigner project. This can be done by:

Run -> Run... -> Select Run time workbench and click on new and then click run.

Creation of a new Run time workbench is needed only for the first run. In subsequent run, just select the newly created configuration can click run.

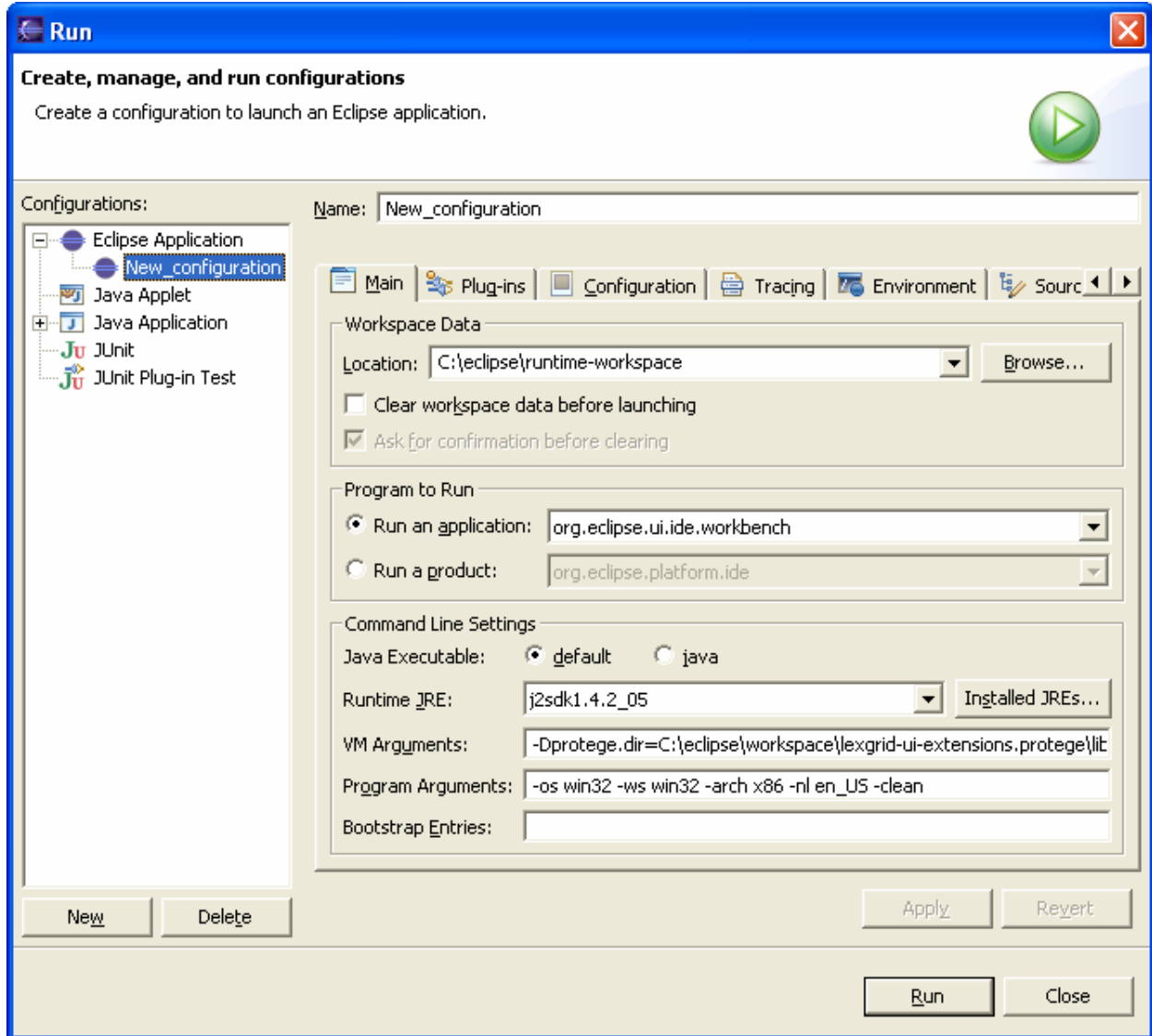


Figure 18: Starting the BPEL Designer project

CREATING A NEW BPEL PROCESS FILE

Create a new project in the runtime workbench from step 2. (Read the Eclipse user guide on instructions on starting up a new project)

To create a new BPEL process file, click on the new project and then New -> Other... This brings up a wizard for a new. In the wizard, expand Examples and then BPEL Process

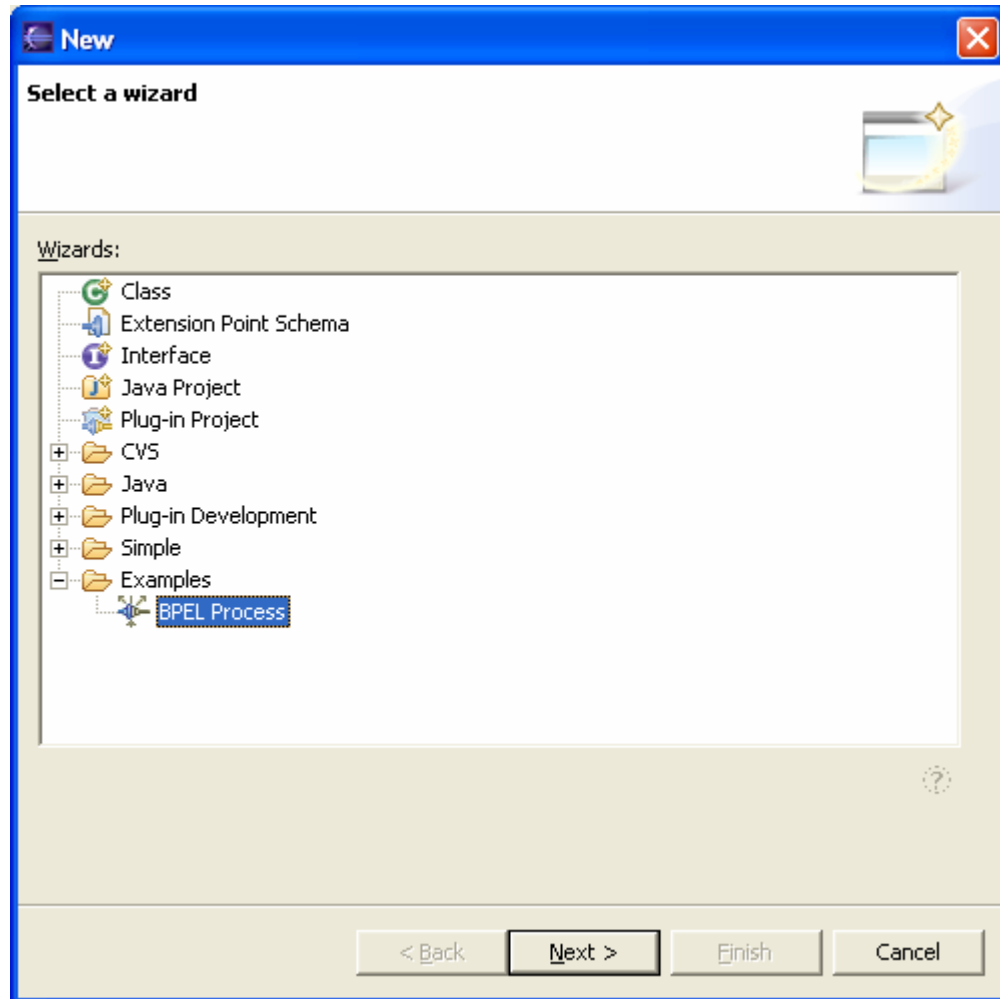


Figure 19: Creating a new BPEL process File

Click on next and specify name file. Click Finish.

A empty process is created and displayed. The user can now modify the process as discussed in the next section.

EDITING THE BPEL PROCESS

The skeleton BPEL process is shown in figure 20 below

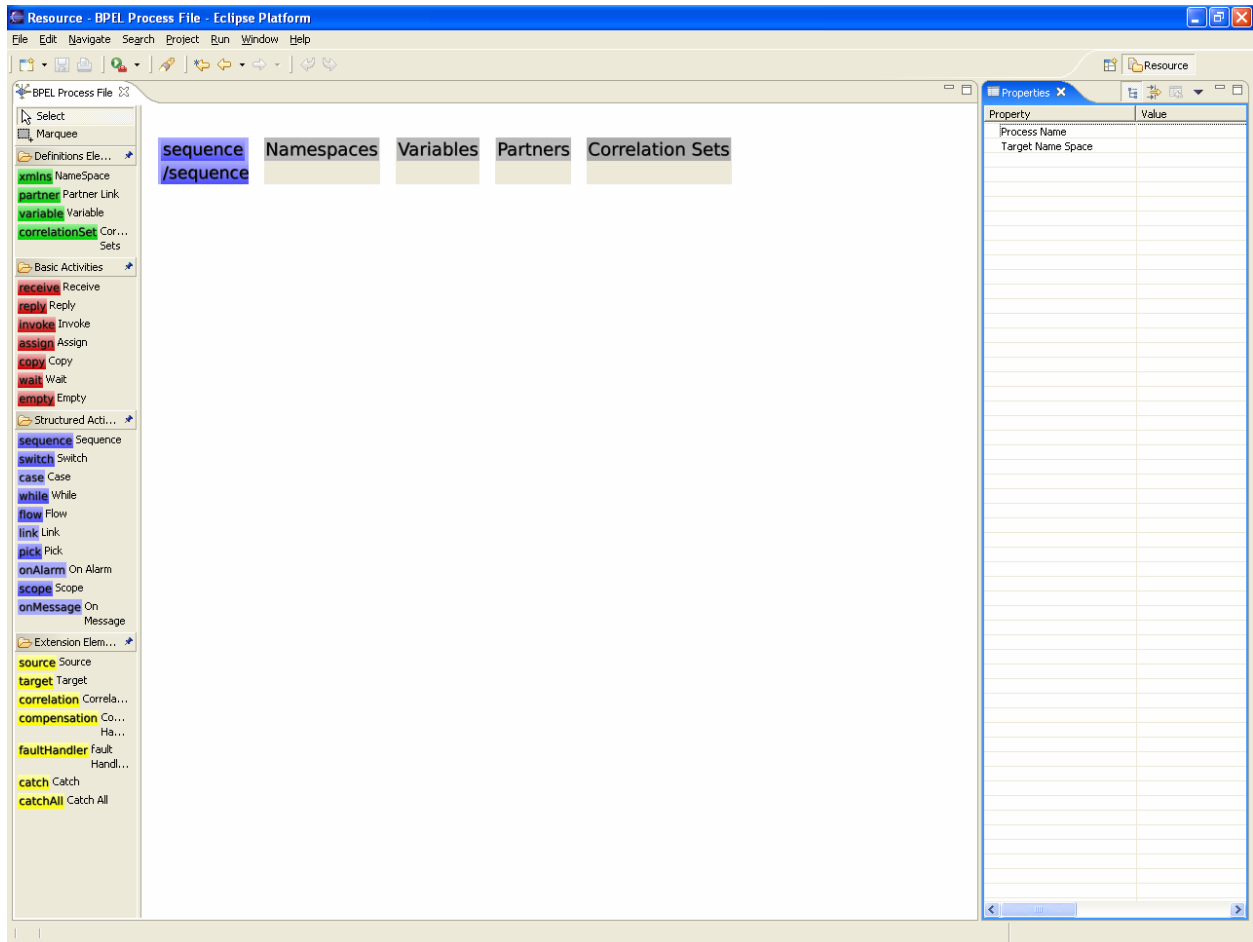


Figure 20: Skeleton BPEL Process

Now the process is ready to be created to work on. All the process elements that can be added to the process are provided as palette entries on the left.

The entries have been categorized according to their functionality and are grouped into corresponding palette groups accordingly. If a palette group is not expanded, it can be done by simply clicking on the group header.

The BPEL designer implements the entire BPEL specification. To know more about functionality of each element, read the latest BPEL specification on OASIS. (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel).

Each element has editable properties that can be set by using the property sheet for the element. To edit properties of elements, click on a element to edit and that will bring up its properties in the property sheet view. If the property sheet is not visible, in the Eclipse menu, select window - > view -> properties. A sample property sheet for a receive element is shown in figure 21 below.

Property	Value
[-] Misc	
Create Instance	Yes
Name	processReceive
Operation	startProcess
Partner Link	caller
Supress Join Failure	No
Variable	processInput
[-] Port Type	
PortType Local Part	startProcessPT
PortType Namespace Prefix	http://tempuri.org/processTNS

Figure 21: Sample propertySheet

To add a BPEL element between two existing elements (in a sequential flow), drag and drop the element on the arrow joining the two elements. When trying to drop on the arrow, the arrow width reduces, indicating that the new element is going to get inserted between the two elements. If dropped anywhere else, the element is added as a new element at the end of the sequence/while or any similar sequential activity. For parallel activities (like flow, pick, switch etc.), the order doesn't matter.

The Designer avoids having ambiguous insertions and hence some actions are not allowed.

Following are some scenarios of disallowed actions

- I. XML namespaces can only be added to namespace container, if dropped anywhere else, it simply doesn't work.
- II. Similarly for partners, correlation sets and variables.
- III. Case can only be contained in a switch.
- IV. Copy can only be contained inside an assign.

Once the process is completed, the process can be saved. On saving a process, a Process WSDL will be created with the same name as the BPEL process file followed by the “.wsdl” extension.

The BPEL file and the process WSDL can now be used to deploy a BPEL process using any available BPEL engine.

A sample WSDL created for a process is shown in Listing 1 below.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:simple:stockQuoteService"
xmlns:tns="urn:simple:stockQuoteService"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:message name="request">
    <wsdl:part name="requestPart" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="response">
    <wsdl:part name="responsePart" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="StockQuotePT">
    <wsdl:operation name="gimmeQuote">
      <wsdl:input message="tns:request"/>
      <wsdl:output message="tns:response"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:service name="simpleBP">
  </wsdl:service>
```

```
<plnk:partnerLinkType ame="StockQuotePLT">
  <plnk:role>
    <plnk:portType name=""/>
  </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="StockQuotePLT">
  <plnk:role>
    <plnk:portType name=""/>
  </plnk:role>
</plnk:partnerLinkType>
</wsdl:definitions>
```

Figure 22: Sample process WSDL

Note: Be default all message types have a single message part which is an xsd String. To have a message as a complex type, the corresponding message needs to be modified in the process WSDL.