A STUDY OF LANGUAGE-ACTION PERSPECTIVE AS A THEORETICAL FRAMEWORK FOR WEB SERVICES

A Thesis in
Information Sciences and Technology
by
Karthikeyan Umapathy

© 2008 Karthikeyan Umapathy

Submitted in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

May 2008
The thesis of Karthikeyan Umapathy was reviewed and approved* by the following:

Sandeep Purao  
Associate Professor of Information Sciences and Technology  
Thesis Adviser  
Chair of Committee

John W. Bagby  
Professor of Information Sciences and Technology

John Yen  
Associate Dean for Research and Graduate Programs  
University Professor of Information Sciences and Technology

Russell R. Barton  
Professor of Supply Chain and Information Systems

Michael D. McNeese  
Professor-in-Charge  
Professor of Information Sciences and Technology

*Signatures are on file in the Graduate School.
Abstract

A science of services is contingent on descriptive and normative theoretical contributions. This dissertation investigates the appropriateness of the language-action framework as a theoretical source for web services, the technological component of a science of services.

Web service supports interoperability among applications independent of their platform, through a set of standards. Current web service efforts are hindered by complexities created by the proliferation of standards which are sometimes competing and/or overlapping. These complexities undermine the usability of web services and therefore hinder widespread adoption. These complexities can be managed by developing boundaries for each web service aspect and new design methodologies and principles that are tailored towards web services. The Language-Action Perspective (LAP) is a candidate theoretical perspective for addressing these concerns because it follows the basic premise of web services—supporting communication among disparate applications to achieve business goals.

The central hypothesis of this dissertation is that the Language-Action Perspective (LAP) would be an appropriate theoretical framework for web services. This hypothesis is tested through three inter-dependent essays. The first essay investigates whether LAP constructs can describe and explain the web services architecture. Findings from this essay reveal a need for conversation mechanisms to guide interactions among web services. The second essay responds to this finding. The domain selected for this investigation is enterprise integration. This essay develops a mechanism for accessing design knowledge with LAP constructs. The outcome is translated into web service conversation specifications. As proof of concept, a research prototype that uses a LAP based mechanism to generate web service conversation specifications as integration solutions is also developed. The third essay assesses the contribution of LAP constructs for designing effective integration solutions. The contribution of LAP constructs were assessed by designing and carrying out a controlled experiment. The assessment shows that LAP based mechanism produces higher quality integration solutions and reduces designer effort required for developing solutions.

Significant contributions from this dissertation include a LAP-inspired reference framework for web services (essay 1), a methodology and software artifact for designing enterprise integration solutions as conversations among web services (essay 2), and an evaluation of the relative effectiveness of LAP constructs for designing web service solutions (essay 3). Together they test the central hypothesis—the appropriateness of LAP as a theoretical framework for web services.
Table of Contents

LIST OF FIGURES ............................................................................................................ vii
LIST OF TABLES ................................................................................................................ viii
ACKNOWLEDGEMENTS ................................................................................................. x

CHAPTER 1 INTRODUCTION ............................................................................................. 1
  1.1 INTRODUCTION ........................................................................................................... 2
  1.2 PROBLEM DEFINITION .............................................................................................. 2
    1.2.1 Theories used in the web service domain .............................................................. 3
    1.2.2 Need for theoretical framework ........................................................................... 5
  1.3 A KEY CHALLENGE ................................................................................................... 7
  1.4 MOTIVATION .............................................................................................................. 7
  1.5 SCOPE ....................................................................................................................... 8

CHAPTER 2 LITERATURE REVIEW .................................................................................. 9
  2.1 WEB SERVICES ......................................................................................................... 10
    2.1.1 Service-Oriented Architecture .......................................................................... 11
    2.1.2 Web Service Technologies ................................................................................. 12
    2.1.3 Service interactions ............................................................................................ 15
    2.1.4 Web service complexity ...................................................................................... 21
  2.2 LANGUAGE-ACTION PERSPECTIVE (LAP) ............................................................... 22
    2.2.1 Speech Act Theory ............................................................................................ 24
    2.2.2 LAP-based applications ..................................................................................... 27
    2.2.3 Criticisms on LAP approach ............................................................................. 36
  2.3 THE SHIFTING PARADIGM AND ITS ISSUES ............................................................ 39
    2.3.1 The LAP approach for web services ................................................................... 40

CHAPTER 3 RESEARCH QUESTIONS AND APPROACH .................................................. 42
  3.1 RESEARCH QUESTIONS ............................................................................................ 43
  3.2 RESEARCH METHODOLOGY ................................................................................... 44
    3.2.1 First Essay: Conceptual Development and Logical Arguments ....................... 44
    3.2.2 Second Essay: Design Science Research Framework ........................................ 45
    3.2.3 Third Essay: Controlled Experiment ................................................................. 47

CHAPTER 4 A THEORETICAL INVESTIGATION OF THE EMERGING STANDARDS FOR WEB SERVICES ..................................................................................... 48
  4.1 INTRODUCTION ......................................................................................................... 49
  4.2 WEB SERVICES AS THE REALIZATION OF SERVICE-ORIENTED COMPUTING .......... 51
    4.2.1 Alternative instantiations .................................................................................... 51
    4.2.2 Commonalities across instantiations ................................................................. 53
  4.3 A THEORETICAL FRAMEWORK FOR THE WEB SERVICES STANDARDS SPACE ........ 55
    4.3.1 A brief review of research on the language-action perspective ......................... 57
    4.3.2 An LAP-inspired framework for the web services standards space .................. 58
  4.4 ASSESSMENT OF EXISTING WEB SERVICES STANDARDS STACKS ......................... 63
    4.4.1 Assessment of the W3C initiative ..................................................................... 63
CHAPTER 5 ACCESSING DESIGN KNOWLEDGE WITH SPEECH ACTS TO GENERATE WEB SERVICE CONVERSATION SPECIFICATIONS ................................................................. 74

5.1 INTRODUCTION .................................................................................................................. 75
5.2 DESIGN KNOWLEDGE ........................................................................................................ 76
5.2.1 Patterns as design knowledge ..................................................................................... 77
5.2.2 Enterprise integration patterns .................................................................................... 78
5.3 ACCESSING ENTERPRISE INTEGRATION PATTERNS ....................................................... 79
5.3.1 Speech acts as mediator ............................................................................................... 80
5.3.2 Components facilitating mapping ............................................................................... 82
5.3.3 Inferring mapping between task and integration patterns with speech acts ............. 86
5.3.4 Surfacing conflicts among integration patterns ......................................................... 89
5.3.5 Demonstration through an example .......................................................................... 90
5.4 ASSEMBLING INTEGRATION PATTERNS INTO WEB SERVICE SOLUTIONS ................. 93
5.4.1 Generating conversation specifications ....................................................................... 95
5.5 RESEARCH PROTOTYPE .................................................................................................. 97
5.6 VALIDATION AGAINST DESIGN SCIENCE RESEARCH GUIDELINES ......................... 100

CHAPTER 6 EMPIRICAL EVALUATION ................................................................................. 103

6.1 NEED FOR EMPIRICAL EVALUATION .............................................................................. 104
6.2 RESEARCH HYPOTHESIS .................................................................................................. 105
6.2.1 Design support for accessing integration patterns ..................................................... 106
6.2.2 Dealing with complexity ............................................................................................ 108
6.3 EXPERIMENTAL DESIGN ................................................................................................ 112
6.3.1 Independent variables ............................................................................................... 112
6.3.2 Dependent variables .................................................................................................. 113
6.4 SUBJECT SELECTION ........................................................................................................ 115
6.5 EXPERIMENTAL PROCEDURES ..................................................................................... 117
6.6 PILOT STUDY .................................................................................................................... 118
6.7 FINAL EXPERIMENT ........................................................................................................ 119
6.8 DATA COLLECTION .......................................................................................................... 120
6.9 ANALYSIS ........................................................................................................................ 121
6.9.1 Assumptions of MANOVA ....................................................................................... 121
6.9.2 Canonical correlation test .......................................................................................... 125
6.9.3 Bivariate correlation test ........................................................................................... 127
6.9.4 MANOVA test for hypothesis 1 and 2 ...................................................................... 128
6.9.5 MANOVA for hypothesis 3 and 4 ............................................................................ 133
6.10 DISCUSSION .................................................................................................................... 137

CHAPTER 7 CONCLUSION ...................................................................................................... 141

7.1 SYNOPSIS ......................................................................................................................... 142
7.2 INFORMATION-TECHNOLOGY-PEOPLE (ITP) PERSPECTIVE ......................................... 144
7.3 CONTRIBUTIONS AND OUTCOMES ................................................................................ 146
7.3.1 Contributions to research ......................................................................................... 147
7.3.2 Contributions to practice ................................................................. 148
7.3.3 Contributions to the I-discipline ......................................................... 148
7.4 Limitations and future research opportunities ................................. 149
7.5 Conclusion............................................................................................. 151

REFERENCES .............................................................................................. 152

APPENDIX A PROPOSED WEB SERVICE STANDARDS .............................. 173
APPENDIX B EXPERIMENTAL INSTRUCTIONS ..................................... 176
APPENDIX C QUESTIONNAIRE............................................................... 186
APPENDIX D BUILDING ENTERPRISE INTEGRATION SOLUTIONS WITH PATTERNS ............................................................. 189
APPENDIX E ENTERPRISE INTEGRATION SCENARIOS.............................. 191
List of Figures

Figure 1.1. Theories used...................................................................................................... 4
Figure 1.2. Challenge in using LAP as theoretical framework for web services .......... 7
Figure 2.1. Service-Oriented Architecture (SOA)............................................................. 12
Figure 2.2. Simple conversation for action (taken from (Winograd & Flores, 1986), pg 65) 28
Figure 2.3. Business Action Theory Framework (modified based on (Goldkuhl, 1998)) ...... 30
Figure 3.1. An overview of the research approach............................................................. 43
Figure 4.1. Comparing alternative instantiations with the travel agent scenario .......... 54
Figure 5.1. Request-Reply pattern..................................................................................... 78
Figure 5.2. Mapping sets of tasks against integration patterns ........................................ 82
Figure 5.3. Enterprise integration patterns as sequence of speech acts............................ 85
Figure 5.4. Conceptual model of the OWL knowledge base ............................................. 87
Figure 5.5. Business process diagram for customer order processing............................. 91
Figure 5.6. Simplified skeleton of ceipXML specification................................................... 95
Figure 5.7. Mapping between ceipXML and cpXML specifications................................. 96
Figure 5.8. Partial ceipXML and cpXML mapping for SCM scenario............................... 96
Figure 5.9. Process for developing integration solutions.................................................... 97
Figure 5.10. Architecture of the Integration Designer Assistant......................................... 98
Figure 5.11. Business process diagram editor interface..................................................... 98
Figure 5.12. Integration Designer Assistant........................................................................ 100
Figure 5.13. Summary of research effort in accordance to design science guidelines....... 102
Figure 6.1. Conceptual model of hypotheses to be tested................................................... 111
Figure 6.2. Plot of ordered squared Mahalanobis distances.............................................. 125
Figure 7.1. Contributions of this dissertation from ITP perspective................................... 145
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 4.1</td>
<td>An LAP-inspired framework for the web services standards space</td>
<td>62</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Assessing the W3C initiative against the LAP-inspired framework</td>
<td>63</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Assessing the Semantic Web Services-initiative against the LAP-inspired framework</td>
<td>66</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Assessing the ebXML-based initiative against the LAP-inspired framework</td>
<td>68</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Assessment of the three standards initiatives against the LAP-inspired framework</td>
<td>72</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Parsimonious set of speech acts</td>
<td>83</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Mapping between speech acts and action types</td>
<td>86</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Partial set of enterprise integration pattern inference rules</td>
<td>89</td>
</tr>
<tr>
<td>Table 5.4</td>
<td>Speech act sequence for identified interactions</td>
<td>92</td>
</tr>
<tr>
<td>Table 5.5</td>
<td>Integration patterns for identified interactions</td>
<td>92</td>
</tr>
<tr>
<td>Table 5.6</td>
<td>Design-Science Research Guidelines (Taken from (Hevner et al., 2004))</td>
<td>101</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>Summary of factors under study</td>
<td>111</td>
</tr>
<tr>
<td>Table 6.2</td>
<td>Design of Experiment for evaluating utility of the LAP constructs</td>
<td>112</td>
</tr>
<tr>
<td>Table 6.3</td>
<td>Design error assessment scheme</td>
<td>115</td>
</tr>
<tr>
<td>Table 6.4</td>
<td>Subject problem and design support assignment groups</td>
<td>120</td>
</tr>
<tr>
<td>Table 6.5</td>
<td>Descriptive statistics</td>
<td>122</td>
</tr>
<tr>
<td>Table 6.6</td>
<td>Descriptive statistics for Box-Cox transformed dependent variables</td>
<td>123</td>
</tr>
<tr>
<td>Table 6.7</td>
<td>Descriptive statistics for LOG transformed dependent variables</td>
<td>124</td>
</tr>
<tr>
<td>Table 6.8</td>
<td>Canonical correlation test results</td>
<td>126</td>
</tr>
<tr>
<td>Table 6.9</td>
<td>Correlation matrix among dependent variables</td>
<td>127</td>
</tr>
<tr>
<td>Table 6.10</td>
<td>Correlation matrix among dependent variables with composite variable</td>
<td>128</td>
</tr>
<tr>
<td>Table 6.11</td>
<td>Descriptive statistics for design support levels as independent variable</td>
<td>129</td>
</tr>
<tr>
<td>Table 6.12</td>
<td>Levene’s test of equality of error variances with design support levels as independent variable</td>
<td>129</td>
</tr>
<tr>
<td>Table 6.13</td>
<td>Effects of design support on design error and design effort</td>
<td>130</td>
</tr>
<tr>
<td>Table 6.14</td>
<td>Univariate test for dependent variables with design support levels as independent variable</td>
<td>131</td>
</tr>
</tbody>
</table>
Table 6.15  Comparison of means for dependent variables with design support levels as independent variable ........................................................................................................................................................................... 132
Table 6.16  Descriptive statistics for design support and problem complexity levels as independent variable ........................................................................................................................................................................... 133
Table 6.17  Levene’s test of equality of error variances with design support and problem complexity levels as independent variable ........................................................................................................................................................................... 135
Table 6.18  Effects of design support and problem complexity on design error and design effort ........................................................................................................................................................................................................................................... 135
Table 6.19  Univariate test for dependent variables with design support and problem complexity levels as independent variable ........................................................................................................................................................................... 136
Table 6.20  Research hypothesis results summary ........................................................................................................................................................................................................................................... 138
Table 7.1  Contributions........................................................................................................................................................................................................................................................................................................... 146
Table 7.2  Opportunities for further research........................................................................................................................................................................................................................................................................................................... 149
Acknowledgements

I am indebted to my parents, friends, and colleagues who have supported and helped me to complete this dissertation. I wish to express my gratitude to everyone who have influenced and contributed to this dissertation, however, I will refrain myself to mention only the most influential persons.

First of all, I would like to thank my dissertation advisor Dr. Sandeep Purao. He has been instrumental in bringing out the best in me through his critical and out-of-box thinking. In the past five years, I have learned a great deal from him about how to be a good scholar and how to express ideas with clarity. Through his expertise in a board domain of information systems, he has helped me to strengthen and broaden my research horizon beyond this dissertation, thus laying a foundation for my future undertakings.

I am grateful to my other committee members Dr. John W. Bagby, Dr. John Yen and Dr. Russell R. Barton. Dr. Bagby has helped me to gain greater understanding of different aspects of standardization processes. Through collaborations with Dr. Yen I learned how to use multi-agent systems for the benefit of web services. I am truly indebted to Dr. Barton who has invested considerable amount of time and energy in guiding me to design experiments and analyze data.

I would like to acknowledge and thank all IST undergraduate students who participated in my research.
A special thanks to Dr. Brain Cameron for allowing me to conduct experiments in his class. I would like to thank all faculties of College of IST and most importantly – Betty, Kathy, Marie, Rhonda and Sue for providing a cordial environment.

I would like to thank Nanda, Samrat, and Nazar for being there when I needed them and always wishing best for me. Thanks to Amie, Annemijn, Haiyan, Joanna, Leslie, Lu, Nick, Thomas, Sharoda, and Umer for making the IST park a fun place to be.
CHAPTER 1  INTRODUCTION

“If the film *The Graduate* were remade today, the word of career advice whispered in Dustin Hoffman's ear might well be ‘services’ instead of ‘plastics’.”

- Paul Horn, senior vice-president of IBM Research (Horn, 2005)

Relevant publications

   
   *Published at*: 2007 Service Computing Conference (SCC)
1.1 Introduction

A distinct shift in the economy has begun since the last industrial revolution that brought us large-scale manufacturing industries. Two-thirds of global economic activity and eighty percent of the United States economy is engaged in the service sector (ABCDoha, 2006). Information technology is the catalyst for this shift (Applegate, 2004; Chesbrough & Spohrer, 2006). This economic shift is pushing organizations to fundamentally transform their businesses, requiring changes to both business models and underlying technology (IBMSummit, 2004; Umapathy, 2007).

A critical technology platform required for coping with this shift is Service-Oriented Computing (SOC) (Papazoglou, Traverso, Dustdar, Leymann, & Krämer, 2005). The fundamental element to develop software applications in the SOC paradigm is “service” (Curbera, Khalaf, Mukhi, Tai, & Weerawarana, 2003). Services are autonomous, platform-independent computational entities which can be combined in numerous ways to achieve business goals. The current realization of the SOC concept is web services (Weerawarana, Curbera, Leymann, Storey, & Ferguson, 2005). They provide interoperability among applications from different platforms and loose coupling of disparate systems through standardized mechanisms (Curbera et al., 2003; Papazoglou & Georgakopoulos, 2003). Web services support interoperability among applications through a set of standards.

1.2 Problem Definition

The efforts of the web services community to develop interoperable standards are coordinated by industrial consortiums (Peer, 2002). These consortiums use processes that were specifically
designed to promote industry consensus and unite disparate web service standardization efforts (Graham, Pollock, Smart, & Williams, 2003). Through these processes, most major software companies heavily influence current web service efforts (Heywood, Jander, Roberts, & Saunders, 1997; Konuru & Mukhi, 2003; Virili, 2003). Therefore, current web service efforts have mainly been ad hoc (Lastra & Delamer, 2006). These efforts focus on a particular set of features without having a complete view of web services (Mukhi, Plebani, SilvaLepe, & Mikalsen, 2004). As a result, these efforts lack holistic view of web services and are not supported by any theoretical framework.

1.2.1 Theories used in the web service domain

I examined 640 research articles published in the proceedings of the International Conference on Web Services (ICWS) and the Services Computing Conference (SCC) in the past three years. This examination was conducted with the objective of identifying theories that are being used to address web service problems. The examination revealed that 57 articles clearly identified theories that the authors used to address specific web service problems. A total of twenty-one theories were applied to address sixteen concerns related to web services, which are listed below. Automata theory, Petri net theory, and Set theory are the most highly used theories in the web service domain. Figure 1.1 shows the percentage of articles for each theory.
This examination reveals that the most web service researchers use formal theories to address specific web service aspects. These formal theories provide systematic approaches to address specific web service problems. This examination strengthens the argument of Mukhi et al. (Mukhi et al., 2004) that current web service efforts focus on specific web service aspects and lack a holistic view. Due to the narrow scope of formal theories they are not appropriate to provide holistic view for web services.
1.2.2 Need for theoretical framework

Without appropriate guidance from a holistic theoretical framework, the advancement of web services has suffered (Balasooriya & Prasad, 2005). A manifestation of this problem can be seen in the significant confusion web service developers’ face when attempting to choose appropriate standards for the different aspects of web service solutions. For instance, WS-Notification (WS-BaseNotification, 2004) and WS-Eventing (WS-Eventing, 2006) are competing standards for notifying the occurrence of an event (Krill, 2006). Without a strong theoretical framework to guide their efforts, developers cannot make informed choices (Hoskin & Harbour, 2005).

Another manifestation of this problem is evident from potential strategic missteps in web service standardization efforts (Festa, 2003). For instance, overlapping standards are being proposed, such as WS-BPEL (WS-BPEL, 2005) and WS-CDL (WS-CDL, 2005), and there is often no separation between standards that address different aspects of web services (Sedighi & Johnson, 2004).

Another manifestation of this problem is the slow adoption of web services (M. Chen, Chen, & Shao, 2003; Ran, 2003; Senf, 2005). Organizations have been slow in adopting web services, since existing development models and tools are inadequate to surmount web service problems (Bosworth, 2001; Papazoglou & Yang, 2002). In order to overcome this drawback, theoretical frameworks for web services need to be developed and would allow development of theoretically informed solutions to web service problems.
Any theoretical framework selected for web services should match with the core principle of supporting communication among disparate applications to achieve business goals (Gottschalk, Graham, Kreger, & Snell, 2002; Papazoglou, 2003). Communication underpins core web service operations such as “publish-find-bind” as well as additional functionalities such as exchanging proposals, negotiating, establishing contract, fulfilling commitments and renewing relationships. A theoretical framework for web services, therefore, should have communication at its core, account for a layered view of the many web service functionalities, and must be directed towards constructing web service solutions that achieve business goals.

Prima facie, these requirements are met by the Language-Action Perspective (LAP). The central premise of LAP is that much work in organizations is performed through language that organizational actors use to achieve their goals (Schoop, 2001; Weigand, 2006). LAP recognizes that language is not only used for exchanging information, (as in reports or statements) but also to perform actions (as in promises, orders, requests, and declarations) (Weigand, 2003). LAP emphasizes that such actions should be the foundation for creating effective information systems. This insight has profound implications for the theory and practice of designing systems using web service technologies to support organizational communication and interaction. LAP, with its essential qualities of communication and coordination, can provide an appropriate framework for web services to address its shortcomings.

Therefore, the hypothesis I investigate in this dissertation is that The Language-Action Perspective (LAP) is an appropriate theoretical framework for web services.
1.3 A Key Challenge

According to LAP, organizations are social systems with actors that communicate to achieve mutually agreed goals (Dietz, 2003). LAP considers information systems to be social systems that are technically implemented to support work in organizations (Goldkuhl & Lyytinen, 1982). As originally formulated, the purpose of LAP is to support human work in organizations (see figure 1.2), in other words, the foundations of LAP are based on human communicative theories. Web services, in contrast, primarily support communication among machines. A major challenge in this research, therefore, is adapting LAP constructs to explain machine to machine communication.

![Diagram](Image)

**Figure 1.2. Challenge in using LAP as theoretical framework for web services**

1.4 Motivation

The motivation of this study is to contribute to the discourse and search for theoretical foundations for web services. As the organizational computing paradigm shifts away from an object-orientation (Booch, 1993; Welke, 1994) to a service-orientation (Papazoglou & Georgakopoulos, 2003), theoretical foundations are necessary to inform design and development of software applications. The development of theoretical foundations for web services has the
potential to greatly accelerate the standardization process, and can provide web service researchers ability to develop more effective web service solutions. In practical terms, stakeholders armed with such a theoretical understanding could explore and evaluate different choices before making heavy resource investments for developing standards or for implementing and testing web service solutions.

1.5 Scope

The scope of this dissertation is to investigate whether LAP is an appropriate theoretical framework for web services. This investigation is conducted as three part research. The first part investigates the ability of LAP to describe and explain web service architecture. The second part demonstrates the ability of LAP to provide solutions to web service problems. The third part evaluates the utility of the LAP based solution developed in the second part. This dissertation does not claim that LAP is the best theoretical framework for web services. The scope of this dissertation does not include comparative tests against other theories.

Following this introduction, chapter 2 provides background and review of prior work on web services and LAP. Chapter 3 describes the research approach and methodology adapted in this dissertation. Chapter 4, 5, and 6 describe the first, second, and third parts of this three-part research, respectively. Chapter 7 provides summary of outcomes and contributions of this dissertation.
CHAPTER 2  LITERATURE REVIEW
This dissertation deals with web services and the Language-Action perspective (LAP), therefore, this chapter provides the required background and a review of prior work related to these two topics. Section 2.1 provides background and review of relevant research on web services. Section 2.2 provides background and review of relevant research on LAP. Section 2.3 provides discussion on some critical problems faced by web services and LAP as a possible theoretical candidate to provide informed solutions to those problems.

2.1 Web Services

The growth of Internet technologies such as eXtensible Markup Language (XML) (Alonso, Casati, Kuno, & Machiraju, 2004), has changed the way business collaborations are supported (Benatallah, Dumas, Fauvet, & Rabhi, 2003). In particular, web services, which are built on top of existing Internet infrastructure, provide an open and XML-based standardized framework for application-to-application interactions.

Web service, following the Service-Oriented Computing (SOC) paradigm, promises to solve many application integration problems. The SOC paradigm provides characteristics of loose coupling and dynamic binding by positioning its basic essence of computing as a “service” (Curbera et al., 2003). Software application (service), developed following a SOC paradigm, defines their functional requirements, nonfunctional requirements and capabilities in an agreed upon, machine-readable format. Services, thus, represent the basic building blocks, that can be combined in particular ways to achieve business goals. Moreover, web service is a collage of standards and technologies (Sleeper & Robins, 2001), that allow applications to communicate with each other remotely, regardless of language or platform it was developed (Manes, 2003).
Thereby, solving problems of tight coupling, hard-coding, and heavy-handed implementations of application integrations.

2.1.1 Service-Oriented Architecture

The web service framework is currently being realized through three web service initiatives: the W3C initiative (W3C-WSActivity, 2005), the Semantic web services initiative (Paolucci & Sycara, 2004) and the ebXML initiative (ebXML, 2005). While these different initiatives may be seen as dissimilar visions of the future of the web service framework, these different directions are merely different instantiations of the basic concept of SOC, called Service-Oriented Architecture (SOA) (Manes, 2003).

The SOA consists of three basic operations—publish, discover, and bind—and three basic participants—the service provider, the service discovery agency, and the service requestor (client) (Manes, 2003), as shown in the figure 2.1. Typically, the service provider develops the service application and a service description file that describes the capabilities of the application. Then the service provider publishes the service description file through a service discovery agency, thus making its service discoverable. The service requestor uses a discover operation to retrieve the service description from the service discovery agency. Next, based on the instructions in the service description, the requestor binds and invokes the provider’s service (Papazoglou & Georgakopoulos, 2003).
2.1.2 Web Service Technologies

Building on the basic SOA principles, web service is evolving to meet the complex needs of integrating distributed business applications over the Internet infrastructure. Web service architecture comprises three key building blocks—service description, service discovery, and service invocation—that are described below.

**Service Description**

The notion of “service” is central to the web service approach (Alonso et al., 2004); therefore, a service provider developing an application (service) using web service must describe what exactly the service does and how it can be accessed (Tsagatidou & Pilioura, 2002). Service providers typically supply details on the various operations available, expected input and output messages including their formats, endpoints for exchanging messages, transportation protocols, and location where service can be accessed (Ferris & Farrell, 2003). These service description details should be published in a manner understandable to the service requestor. Service
providers can use standardized mechanisms known as web service description specifications, for publishing their service descriptions in a complete and comprehensible way. With respect to the W3C initiative, service providers can use Web Services Description Language (WSDL) (WSDL, 2001) as the standard for describing a service. With respect to the semantic web services initiative, service providers can use Web Ontology Language for Web Services (OWL-S) (OWL-S, 2005) as the standard for describing a service. With respect to the ebXML initiative, service providers can use Business Process Specification Schema (BPSS) (BPSS, 2001) as the standard for describing a service.

Service Discovery
Another key function of web service is to let service providers advertise their service descriptions in a manner discoverable by interested service requestors (Alonso et al., 2004). For this purpose, providers use a service registry to advertise their services and requestors use the service registry to look up services that fit their interests (Tsagalatidou & Pilioura, 2002). Service providers require appropriate mechanisms to register and store their service descriptions and the associated details (Gottschalk et al., 2002). Service requestors require appropriate mechanisms to search for and locate needed services (Gottschalk et al., 2002). Typically, these registries can be either publicly available for everyone, or privately owned and controlled by individual organizations or consortiums (Alonso et al., 2004). Standardized mechanisms known as service discovery specifications provide this key aspect. With respect to the W3C initiative, service discovery is facilitated through Universal Description, Discovery and Integration (UDDI) (UDDI, 2004). With respect to the semantic web services initiative, there are no recognized
mechanisms available. With respect to the ebXML initiative, service discovery is facilitated through the ebXML Registry (ebRS, 2002).

**Service Invocation**

After finding a service description specification that fits his or her interest through a service discovery mechanism, the service requestor must be able to bind and invoke the service provider application (Alonso et al., 2004). For this purpose, both the service provider and the service requestor must use common messaging protocol to exchange messages containing the appropriate service calls. This communication is facilitated through Simple Object Access Protocol (SOAP) (SOAP, 2003). All three web service initiatives use the SOAP standard for facilitating message exchanges.

The above three key aspects—service description, service discovery, and service invocation—allow web service to take advantage of SOA. However, in addition to these basic aspects, more elements are required in order to conduct business activities using web service. For instance, service descriptions provide ways to describe only functional details of the service, while non-functional details such as pricing, reliability, and privacy are not captured (Tsalgatidou & Pilioura, 2002). Service discovery mechanisms provide only key words based discovery. Current service discovery mechanisms do not address issues of quality of services (Ran, 2003), ability to describe service requestor interests (SWS Arch, 2005), or service provider trustworthiness (Maximilien & Singh, 2004). Service invocation facilitates message exchanges among services; it does not provide the required protocols for coordinating and collaborating business interactions.
to achieve business goals (Alonso et al., 2004). The next section contains a review of web service protocols that can be used for coordinating and collaborating business activities.

2.1.3 Service interactions

Although service description, service discovery, and service invocation provide the required foundations for simple application integration, they do not provide adequate support for complex business processes (Papazoglou, 2003). A business process can be defined as a series of interrelated activities that are logically sequenced towards achieving a business goal that creates value for customers (Axenath, Kindler, & Rubin, 2005; Chang, 2006). Typically, an activity can be as simple as sending or receiving a message, or executing another process, or it encodes certain business rules (Axenath et al., 2005). Therefore, business processes contain complex interactions involving the execution of operations in a systematic way. These complex interactions often involve invocations of more than one service, therefore the above three key web service aspects are not sufficient to support complex business processes (Alonso et al., 2004; Papazoglou, 2003).

Complex business processes can be supported by providing protocols for (i) aggregating individual web services and executing them according to the appropriate order described in the business process; and (ii) managing business interactions that are oriented towards specific business goals as well as maintaining the needed context to move forward from one service to another (Alonso et al., 2004; Papazoglou, 2003). The above required protocols are supported by web service in two key ways: through service composition and through service conversation.
Service composition

Service composition enables the development of complex services by assembling both elementary and complex services in a particular order (Casati & Shan, 2001). If service providers can identify individual services that perform various activities described in the business process, service composition provides potential for developing complex services by composing individual services according to the business logic described in the business process (Alonso et al., 2004). Thus, service composition provides the ability to develop complex services that support business processes on top of core web service architecture (Milanovic & Malek, 2004), in other words, publish-find-bind.

Service composition is facilitated through three main components: the composition specification, the development environment for developing composition specification, and the run-time environment to execute composition specification (Alonso et al., 2004). The first component, service composition specification, provides required business process modeling language constructs tailored to web service needs for capturing the business logic of a composite web service. The composition specification also details services involved in the composition (Staab et al., 2003). Many such languages, known as web service composition languages, are being developed by different vendors and coalitions. Web service composition languages include Web Service Choreography Interface (WSCI) (WSCI, 2002), Business Process Modeling Language (BPML) (BPML, 2002), Business Process Execution Language for Web Services (BPEL4WS) (BPEL4WS, 2003) that was developed as a part of W3C initiative, XML Process Definition Language (XPDL) (XPDL, 2005) and Business Process Specification Schema (BPSS) (BPSS, 2001) that was developed as a part of the ebXML initiative. A comparative evaluation of these
languages conducted by Wohed et al. against a set of constructs used for composing workflow and communication patterns suggests that BPEL4WS (BPEL4WS, 2003) performs best. BPEL4WS is currently known as Web Services Business Process Execution Language (WS-BPEL) (WS-BPEL, 2005) is a recommended standard of the Organization for the Advancement of Structured Information Standards (OASIS) consortium.

The second service composition component provides support tools for developing web service composition specifications. Typically, these tools contain a graphical user interface with the ability to diagrammatically describe the assemblage of services along with the business logic of the composition. These tools also translate those graphical representations into appropriate web service composition specifications. Some examples of tools for generating WS-BPEL specifications are ActiveBPEL Designer (ActiveBPEL, 2007), Intalio Designer (Intalio|Designer, 2007), and Oracle BPEL Process Manager (OracleBPEL, 2006).

The third service composition component provides a run-time environment for invoking individual services involved in the composition in accordance to the business logic of the composite service. These run-time environments are known as web service composition engines. Some examples of composition engines for executing WS-BPEL specifications are ActiveBPEL Open Source Engine (Engine, 2007), Intalio Server (Intalio|Server, 2007), and Oracle BPEL Process Manager (OracleBPEL, 2006).

The above three components provide the required infrastructure for implementing service compositions. However, several key questions are not answered and these questions are reflected
in the ongoing discussion in the web service community (Alonso et al., 2004; Pfadenhauer, Dustdar, & Kittl, 2005). Some of the questions are:

- How to develop a composition model that describes composite service?
- How to analyze such composition models?
- How to verify such composition models syntactically and semantically? and
- How to test the functionality of such composition models?

Current research efforts utilize various techniques such as UML activity diagrams, Petri-nets, state-charts, rule-based orchestration, activity hierarchies, \(\pi\)-calculus, automata, and process algebra to address the above questions. The following works presented surveys of ongoing service composition research efforts using these techniques: (Alonso et al., 2004), (Beek, Bucchiarone, & Gnesi, 2006), (Dustdar & Schreiner, 2005), (Jinghai & Xiaomeng, 2005), and (Srivastava & Koehler, 2003).

**Service conversation**

Service composition enables the ordered sequencing of service invocation; however, it does not support long-running business conversations that typically occur during the execution of complex business processes (Alonso et al., 2004). These long-running business conversations are critical for the performance and coordination of business processes that involves multiple participants and/or cross organizational boundaries (Papazoglou, 2002). These business conversations can be thought of as a type of discourse, in other words, a sequence of correlated message exchanges among participants (Goldkuhl & Lind, 2004). It is critical for web services to
support such long-running conversations in order to be viable solutions for supporting complex business processes (Papazoglou, 2002).

Service conversation supports long-running conversations through mechanisms that provide a loosely coupled, peer-to-peer interaction model where services exchange several messages (Ardissono, Goy, & Petrone, 2003; Peltz, 2003). Services exchange a series of messages within an explicit conversational context that is established upon initial contact, maintained for the duration of the conversation, and discarded at the end (Hanson, Nandi, & Kumaran, 2002). Each new message in a conversation is interpreted in relation to the messages previously exchanged in that conversation. Thus, conversations among services are represented as multi-step exchanges of correlated messages (Ardissono et al., 2003).

Service conversation is facilitated through two main infrastructure components: a conversation specification and a run-time environment that dispatches messages and maintains the conversational context in accordance with the conversation specification (Alonso et al., 2004; Benatallah, Casati, Toumani, & Hamadi, 2003). The first component, service conversation specification, provides a standardized way of describing interactions among services as an ordered sequence of message exchanges geared towards a business goal. Currently, there are three service conversation specifications: Web Services Choreography Description Language (WS-CDL) (WS-CDL, 2005), Conversation Policy for XML (cpXML) (cpXML, 2002) and Web Services Conversation Language (WSCL) (WSCL, 2002). A comparative evaluation of these specifications against communicative constructs required for supporting long-running conversations suggest that cpXML is more effective than the other two (Umapathy, 2006). The
second component provides mechanisms that facilitate the execution of conversations in accordance with the service conversation specification. Mechanisms should be able to dispatch messages, receive messages, and process each message depending upon the conversation state information (Alonso et al., 2004). Currently, two such mechanisms are available: Conversation Controller (Kuno & Lemon, 2001) and Conversation Support for Web Services (CS-WS) (Hanson, Nandi, & Kumaran, 2002; Hanson, Nandi, & Levine, 2002).

While these two components facilitate the execution of service conversations, there are several important challenges that need to be addressed. Some of the key challenges and open research questions with respect to service conversations are:

- How to develop a conversation model that represents interaction logic for services involved in a composite service? (Benatallah, Casati et al., 2003; Zirpins, Lamersdorf, & Baier, 2004),
- How to make such conversation models complete and comprehensible? (Zirpins et al., 2004), and
- How to verify the correctness of such conversation models? (Yi & Kochut, 2004).

Additionally, recent surveys on state-of-the-art technologies for modeling complex business interactions suggest that there is lack of appropriate methodologies and tool support for designers to generate web service conversation specifications (Lippe, Greiner, & Barros, 2005; Umapathy & Purao, 2006).
2.1.4 Web service complexity

Web service through key elements such as service composition and service conversation permit the integration of applications irrespective of their development platform. This platform-independence has made web service the leading choice of technology for integrating applications (Iyer, Freedman, Gaynor, & Wyner, 2003). However, despite this growing interest and recent efforts, web services largely remain an immature technology (Benatallah, Casati, & Toumani, 2004; Tilley et al., 2002). The current web services architecture is confronted with several critical challenges, such as protecting services from attacks (J. Xu & Lee, 2003), approaches for developing reliable composite service (Dustdar & Schreiner, 2005), approaches for developing complete and comprehensible service conversations (Zirpins et al., 2004), providing consistent and reliable transactional support (Papazoglou, 2002), and the ability to describe non-functional aspects of service (Tsagatidou & Pilioura, 2002). As a result, developing web services is largely a complex activity (Benatallah et al., 2004; Papazoglou & Yang, 2002).

Moreover, there are numerous competing standards for several important functionalities such as business process automation, security, reliable messaging, and coordinating business activities (Leavitt, 2004). The proliferation of competing standards addressing similar functions creates fragmentation and confusion around efforts to standardize web services, instead of aiding the common goal of developing interoperable standards (Alonso et al., 2004; Koch, 2003; Tsagatidou & Pilioura, 2002). These competing standards also cause confusion and delays in designing solutions, developing applications, and mitigating risks (Hoskin & Harbour, 2005; Koch, 2003).
The above two complexities, the difficulties surrounding development of web service applications and the proliferation of standards that sometimes compete, severely undermines the usability of web services and therefore hinders its widespread adoption (M. Chen et al., 2003; Ran, 2003; Senf, 2005). These complexities can be managed by developing boundary conditions for each aspect of web service, design methodologies, and new business models and principles that are tailored towards web services (Karastoyanova, 2004; Papazoglou & Yang, 2002). Development of such methodologies, boundary conditions, models, and/or principles require substantial guidance from a theoretical framework (Amit & Zott, 2001; Markus, Majchrzak, & Gasser, 2002; Walls, Widmeyer, & Sawy, 1992). The next section reviews one such plausible theoretical candidate for web services—the Language-Action Perspective (LAP).

2.2 Language-Action Perspective (LAP)

Through their article, “Doing and speaking in the office,” Flores and Ludlow challenged the conventional notion that communication is merely the transmission of information or symbols and argued that people are linguistic beings who use language to perform actions (Flores & Ludlow, 1980). Through this article, they provided awareness and relevance of communication theories for the information systems field. Goldkuhl and Lyytinen (Goldkuhl & Lyytinen, 1982) coined the term “Language Action View” to describe an approach for designing information systems from the perspective of how people use communication to perform actions. Building on this perspective, Winograd and Flores (Winograd & Flores, 1986) presented a new foundation for designing information systems by conceptualizing actions performed through communications as recurrent communicative patterns. The revolutionary work of Winograd and Flores inspired a wave of diverse Language-Action Perspective (LAP) based applications in the
last two decades (Weigand, 2006). They all have in common the fundamental agreement that language is not only used for exchanging information, as in reports or statements, but also to perform actions such as promises, orders, declarations, etc (Schoop, 2001; Weigand, 2003). LAP emphasizes that such actions should be the foundation for creating effective information systems.

In contrast, traditional approaches consider information systems as repositories for storing representations of facts about the real world, that reflect reality (Yetim & Bieber, 2003). According to these approaches, the important goal of information systems is to process stored facts and provide required information for managerial and decision making purposes (Connors, 1992; Davis & Olson, 1984). Therefore, information systems development is considered a process of manipulating information to meet the requirements of a specific business task (De Michelis et al., 1997). Moreover, requirements for developing systems were based on simplified assumptions and heuristics that capture known properties of the real world while ignoring unknown properties (Oreskes, Shrader-Frechette, & Belitz, 1994). Thus, traditional information systems are seen as ‘mirrors of reality’, where users are provided with abstractions of the reality (Flores, Graves, Hartfield, & Winograd, 1988; Goldkuhl & Lyytinen, 1982). Therefore, each user has a ‘local view’ of the real world, that is the individual’s slice of the reality seen through an information system (Goldkuhl & Lyytinen, 1982). Several researchers within the information systems field have challenged this notion of information systems as an image of reality (Goldkuhl & Ägerfalk, 2000; Hirschheim, Klein, & Lyytinen, 1995; Winograd & Flores, 1986).

On the other hand, the LAP approach presumes that the purpose of an information system is to support communication among people to help them perform actions together (Flores et al., 1988;
Goldkuhl & Lyytinen, 1982). LAP considers communication to be a form of action performed by the participants (Winograd, 2006). Therefore, LAP recognizes the importance of communication in an organizational context and focuses on how communicative aspects are used for performing business actions (Mulder & Reijswoud, 2003). Thus, according to the LAP approach, people are part of a community, who interpret the world and coordinate their actions together in that world (Goldkuhl & Lyytinen, 1982). Thus, the user is seen as a participant in the community of interpretation and information is contextualized for a community of interpreters (Goldkuhl & Lyytinen, 1982).

LAP emphasizes how people communicate with others; how language is used to create a common shared reality and how people use communication to coordinate their activities (Schoop, 2001). Therefore, LAP is grounded in the linguistic and social rules that govern the use of the language. The main theoretical foundation for the LAP approach is the Speech Act Theory (Austin, 1962; Searle, 1969). However, the LAP approach is also influenced by the Theory of Communicative Action (Habermas, 1984), Conversation Analysis (Sacks, 1995) and Organizational Semiotics (Stamper, 1996). The next section provides an overview of the Speech Act Theory.

2.2.1 Speech Act Theory

Speech Act Theory was first introduced by Austin (Austin, 1962), and further developed and formalized by Searle (Searle, 1969). The underlying theme of the theory is that the use of language is not only to describe a situation or fact, but also to perform certain kinds of actions (Goldkuhl & Ågerfalk, 2000). For instance, the utterance of the statement, “You passed the test,”
is considered the performance of a declarative action as opposed to making a statement that may be judged true or false. Thus, according to speech act theory, speaking is acting and by speaking, the speaker performs a ‘speech act’ (Bach & Harnish, 1979).

A speech act is the basic unit of communication that expresses the intention of the speaker, such as making a promise or asserting a claim (Aramaki, Lehtinen, & Lyytinen, 1988). The meaning of a speech act is understood based on the propositional content and context of its occurrence that includes the speaker and hearer, the time and place, and other factors relevant to the performance of communication. The main characteristic of speech act theory is that every speech act consists of four levels of action. Suppose that a speaker succeeds in saying something to a hearer in a given context, then the following acts can be distinguished (Austin, 1962; Moore, 2001):

1. **Utterance act** is the act of uttering something. Utterance act refers to the action of the speaker uttering something to the hearer in the given context.

2. **Locutionary act** is the act of saying something meaningful. Locutionary act refers to the actual action of the speaker saying something to the hearer in the given context.

3. **Illocutionary act** is the act of doing something in saying it. Illocutionary act refers to the action of doing something by the speaker in the given context in virtue of having performed an utterance act.

4. **Perlocutionary act** is the act of affecting the hearer by saying it. Perlocutionary act refers to the effect on the hearer’s feelings, thoughts, or actions in virtue of the speaker having performed the illocutionary act in the given context.
Of the above four acts, the illocutionary act is critical for the successful performance of the speech act since it expresses the communicative intent of the speaker (Traum, 1999). The illocutionary act can be further decomposed into (i) the illocutionary force, that specifies the type of action and (ii) the propositional content, that specifies the details of action (Austin, 1962). The illocutionary force represents the speaker’s intention in producing an illocutionary act. An illocutionary force is the combination of the basic purpose of a speaker in making an utterance, including particular presuppositions, and the attitudes of the speaker (Searle & Vanderveken, 1985). For example, if the speaker says, “You have passed your defense,” then the speaker is declaring to the hearer that he or she passed defense. Thus, in this statement the speaker’s intention is to make a declaration. However, if the speaker says, “Have you passed your defense,” then it can be considered that the speaker is asking a question to the hearer. Therefore, in this statement the speaker’s intention is to ask a question. Thus speakers can perform different actions with the same proposition.

Bach and Harnish (Bach & Harnish, 1979) developed six major categories of speech acts based on the correlation between illocutionary acts and illocutionary forces. These categories of speech acts are constatives, directives, commissives, acknowledgments, effectives, and verdictives. Constatives represent the speaker’s beliefs, intentions, desires, or experiences. Directives represent the speaker’s attempt to get the hearer to perform the action indicated in the propositional content. Commissives represent the speaker’s intention to perform the action indicated in the propositional content. Acknowledgements represent the speaker’s feelings or psychological attitudes regarding the state of affairs represented by the propositional content. Effectives represent the speaker’s effects to change the state of affairs of an institution.
Verdictives represent the speaker making an official judgment relevant to the institutional state of affairs that is social binding in the given context. Bach and Harnish have developed about fifty illocutionary forces based on the above speech act classifications.

Speech act frameworks and classifications can be used for decomposing communicative messages into smaller entities (Kimbrough & Moore, 1997). Thus they provide good foundations for interpreting and representing communicative messages (Auramaki et al., 1988). Speech acts can also be used to create larger wholes such as well-formed discourses and conversations (Hirschheim, Iivari, & Klein, 1997). For instance, if the speaker asks a question to the hearer, the hearer may follow up with an answer or a request for clarification, a refusal to answer or a counter-question. Thus set of speech acts can be ordered in a sequence to form a logical whole (Auramaki et al., 1988).

2.2.2 LAP-based applications

The LAP approach is not merely a philosophical framework but has stirred the development of a number of methodologies and computer based tools for solving specific organizational problems (Lyytinen, 2004; Weigand, 2006). The LAP approach has been applied in diverse fields such as computer supported co-operative work (CSCW), workflows, business process modeling, business process re-design, e-commerce, electronic negotiations, software agents, and virtual communities (Lyytinen, 2004; Weigand, 2006). I provided a review of several LAP-based applications in this section.
**Coordinator**

The Coordinator is one of the earliest applications of the LAP approach (Schoop, 2003). This application was developed based on the observation by Winograd and Flores (Winograd & Flores, 1986) that in the most work environments the coordination of activities are of central importance and these coordinated activities are facilitated by conversations among participants (Beeson & Green, 2003). Winograd and Flores named these conversations “Conversations for Action” (Winograd & Flores, 1986). Through the coordinator application, they provide a mechanism that utilizes a network of speech acts to model conversations that are directed towards explicit cooperative action (Winograd & Flores, 1986). This mechanism uses circles to represent a possible state in the conversation, heavy circles for termination states, and lines connecting the states to represent speech acts that indicate actions taken, see figure 2.2 for an example of a simple conversation for action.

![Simple conversation for action (taken from (Winograd & Flores, 1986), pg 65)](image)

**Figure 2.2.** Simple conversation for action (taken from (Winograd & Flores, 1986), pg 65)
Speech-Act-based office Modelling aPprOach (SAMPO)

The Speech-Act-based office Modelling aPprOach (SAMPO) provides a methodology for modeling organizational information systems (Auramaki et al., 1988). The SAMPO approach considers the purpose of information systems development is to support communication among people to perform their activities (Lyytinen, Lehtinen, & Auramäki, 1987). Therefore, the SAMPO approach considers the process of modeling organizational information systems as a process of modeling organizational discourses (Auramaki et al., 1988). A network of speech acts can be arranged in systematic order to model a discourse that establishes, controls, and coordinates organizational activities (Lyytinen et al., 1987). SAMPO provides a methodology and a graphical tool for modeling complete and coherent organizational discourses by logically structuring the sequence of speech acts (Auramaki et al., 1988).

Business Action Theory (BAT)

Business Action Theory (BAT) (Goldkuhl, 1996, 1998) provides a generic framework for describing business interactions between the customer and the performer. The framework is divided into six phases (Goldkuhl, 1998); (1) Business prerequisites phase, (2) Exposure and contact search phase, (3) Contact establishment and proposal phase, (4) Contractual phase, (5) Fulfillment phase, and (6) Completion phase. The BAT-model emphasizes the idea that a business action involves both material and communicative actions; however, it presumes that material actions can be characterized as communication (Goldkuhl, 1996). The framework provides a graphical model to describe the inherent business logic and business processes as communicative exchanges, with exception to phase 1 (Goldkuhl, 1998). As shown in the figure 2.3, phase 2 involves the exchange of interest, phase 3 involves the exchange of proposals, phase
4 involves the exchange of commitments, phase 5 involves the exchange of values, and phase 6 involves the exchange of claims or acceptance (Goldkuhl, 1998).

Dynamic Essential Modelling of Organizations (DEMO)

Dynamic Essential Modelling of Organizations (DEMO) (Dietz, 1994, 2001) is an information systems development methodology that offers concepts and modeling techniques for re-engineering business processes. According to DEMO, an organization has three levels: documental, informational, and essential (Dietz, 1994). At the documental level, actions such as gathering, storing, transmitting and reproducing information are performed. At the informational level, actions such as deducting or deriving information are performed, preferably using computers. At the essential level, business transactions are executed by human actors. Actions
performed at the essential level are considered authentic business actions, while actions performed at the documental and informational levels are categorized as supporting actions (Dietz, 1994). Therefore, business transactions performed at the essential level are the core of the DEMO methodology and are typically performed by two actors: the initiator and the executor (Dietz, Rijst, & Stollman, 1996). Business transactions are divided into three phases under DEMO: the Order (O) phase, the Execution (E) phase, and the Result (R) phase (Dietz, 2001; Dietz et al., 1996). In the O-phase both actors reach an agreement about the execution of some future action. Then, in the E-phase, the executor performs the negotiated future action. During the R-phase, both actors come to agreement on the facts that were accomplished as a result of the execution of the negotiated future action.

Layered Pattern Approach

Weigand et al. (Weigand, Heuvel, & Dignum, 1998) argues that in electronic commerce business transactions, participants would interact with each other electronically in multiple forms. However, traditional Electronic Document Interchange (EDI) protocols, that provide implementations for converting paper forms to electronic versions, cannot provide adequate support for e-commerce transactions (Weigand et al., 1998). Weigand and Heuvel, building on the notion of pattern analysis, proposed a layered pattern approach for conducting analysis on e-commerce business transactions (Weigand & Heuvel, 1998). This approach consists of five layers, that are the speech act, the transaction, the workflow loop, the contract, and the scenario (Weigand & Heuvel, 1998). The speech act is used to represent intentional action performed through message exchange in the transaction (Weigand & Heuvel, 1998; Weigand et al., 1998). Thus, speech acts are the basic unit of analysis in this approach. Transactions are considered the
smallest sequence of possible actions constituting a business interaction (Weigand et al., 1998). A transaction is formed by compiling a set of speech acts (Weigand & Heuvel, 1998). Workflow loops are sets of related transactions aimed towards a goal. Therefore, a set of transactions are grouped to form a workflow loop (Weigand & Heuvel, 1998). Contracts represent obligations and authorizations among business partners (Weigand et al., 1998). A contract is established by interrelating two or more workflow loops (Weigand & Heuvel, 1998). Scenarios are used to describe multiple interactions across contracts that run concurrently among multiple parties (Weigand et al., 1998). A set of related contracts form a scenario (Weigand & Heuvel, 1998).

**Generic Layered Pattern Approach**

Lind and Goldkuhl (Lind & Goldkuhl, 2001) critically analyzed each layer of the layered pattern approach proposed by Weigand et al. (Weigand & Heuvel, 1998) to show their shortcomings. Lind and Goldkuhl proposed an alternative framework, called the Generic Layered Pattern Approach for any kind of inter-organizational electronic business transaction (Lind & Goldkuhl, 2001), unlike the layered pattern approach that focuses only on e-commerce transactions (Weigand & Heuvel, 1998). The Generic layered pattern approach consists of five layers: the business act, the action pair, the exchange, the business transaction, and the transaction group (Lind & Goldkuhl, 2001). Except for the business act, each layer is derived based on its preceding layer (Lind & Goldkuhl, 2001). A business act can be a communicative and/or a material action (Goldkuhl & Lind, 2002). An action pair represents the grouping of business acts as patterns of triggers and responses (Lind & Goldkuhl, 2001). An exchange constitutes one or more action pairs representing the contribution of one actor in return of the other actor’s contribution (Goldkuhl & Lind, 2002). A business transaction represents a pattern built from...
different types of exchanges related to each other (Lind & Goldkuhl, 2001). Business transactions can be thought of as having different phases involving exchanges that lead to states that satisfy the needs of both the customer and the performer (Goldkuhl & Lind, 2002). A transaction group represents recurrent business transactions framed for a long-term agreement aimed at establishing relationships (Lind & Goldkuhl, 2001).

**Atoms, molecules, and matters**

Analogous to the three layers distinguished in Physics, Dietz suggests three conceptual layers to understand business processes through (Dietz, 2003). Compared to the layered pattern model (Weigand & Heuvel, 1998), that focuses on electronic commerce, and the generic pattern model (Lind & Goldkuhl, 2001), that focuses on inter-organizational interactions; Dietz’s ‘atoms, molecules, and fibers’ model focuses on both inter and intra-organization interactions (Dietz, 2002, 2003). This model is comprised of three layers, atoms, molecules, and fibers, that capture the business of an organization(Dietz, 2003). Atoms consist of coordination acts and action rules (Dietz, 2003). Coordination acts are facilitated by communicative actions. By performing coordination acts, individuals enter into and comply with commitments directed towards achieving material actions. Constraints on ‘Atoms’ are provided by action rules, that contain possible acts and how the choice about actions should be made. Molecules consist of transactions and actor roles (Dietz, 2003). Transactions represent finite sequences of coordination acts between two actors directed towards achieving common objectives. Actor roles define the roles played by actors as sets of action rules. Fibers consist of business processes that are considered to be compositions of interconnected transactions (Dietz, 2003).
Agent communication languages

A software agent is a computer system that acts autonomously on behalf of a user or a program in a given environment in order to meet its design objectives (Nwana, 1996; Wooldridge, 2002). To be useful, an agent should be able to interact with its environment, user, and/or other agents (Wooldridge, 2002). Agent interactions are very complicated because agents need to maintain and share their knowledge, goals, and commitments (Labrou, Finin, & Peng, 1999). In order to maintain and use knowledge, beliefs, and intentions, agents typically exchange messages through certain agreed knowledge representation languages, collectively referred to as agent communication languages (Labrou et al., 1999). There are two well known and widely used languages, the Knowledge Query and Manipulation Language (KQML) (Finin, Labrou, & Mayfield, 1994) and the Foundation for Intelligent Physical Agents—Agent Communication Language (FIPA-ACL) (FIPA-ACL, 2002). Both languages were developed based on the speech act theory, following the fundamental philosophy of language as action (Labrou et al., 1999). These languages used speech act theory to develop appropriate semantics that allow agents to affect other agents’ knowledge, beliefs, and intentions (Pitt, Guerin, & Stergiou, 2000). These languages prescribe speech act based message exchanges to facilitate agents to perform actions upon receiving messages (Colombetti & Verdicchio, 2002).

Formal Language for Business Communication (FLBC)

A main problem in the field of business-to-business electronic commerce is the lack of formal schemes to encode messages for business communication (Weigand & Hasselbring, 2001). Current primary schemes, such as EDIFACT, are too costly, do not provide the required expressiveness, and are not flexible enough to cope with the dynamics of the new economy.
(Weigand et al., 1998). Moore developed the Formal Language for Business Communication (FLBC) based on speech act theory (Moore, 2001). FLBC defines broad range message types that express the speaker’s intentions while distinguishing those intentions from type of message content. FLBC utilizes the illocutionary force from speech act theory to provide separation between message types and message contents (Weigand & Hasselbring, 2001). This clear distinction makes it easier for e-commerce applications to interpret and respond to messages (Moore, 2001). FLBC message syntax was developed using XML (XML, 2006), a markup language for self-descriptive data.

Other applications based on the LAP approach that were developed based on inspiration provided by the above applications are mentioned here. eXtensible Language for Business Communication (XLBC), drawing on FLBC (Moore, 2001), semantic network of the multilingual thesaurus and layered pattern approach (Weigand & Heuvel, 1998) provides an extensible communication infrastructure that permits message exchanges to be grouped into different aggregation levels of conversation (Weigand & Hasselbring, 2001). Milano provides a system for supporting multimedia conversations within a work process (De Michelis & Grasso, 1994). Action Technologies extended the Conversation for Action concept (Winograd & Flores, 1986) to develop a business modeling approach known as the Action Workflow Loop, that consists of four basic steps—proposal, agreement, performance, and satisfaction (Medina-Mora, Winograd, Flores, & Flores, 1992). Negoisst is a negotiation support system for conducting complex electronic negotiations over the Internet using both structured message exchanges and cooperative document exchanges (Schoop, Jertilä, & List, 2003). Negoisst used the LAP approach to create its architecture and to structure message and document exchanges (Schoop,
2003). Johannesson provides a speech act theory based information systems development framework that reconciles the representation-based and communicative-based approaches to systems development (Johannesson, 1995). This framework shows how to model communications as discourses, and, at same time, view discourses as sequences of events. Cooperative Information Agents (CIA) is a communication system for the business-to-business application domain that allows software agents to query for required information and also to negotiate based on the information the agent can provide (Verharen, Dignum, & Weigand, 1996). CIA uses speech act theory to model message exchanges among agents.

Regardless of the continued development of diverse LAP-based applications, the LAP computing movement has not penetrated mainstream organizational computing (Lyytinen, 2004). For example, LAP concepts are not taught in most universities or in computing textbooks (Schoop & Kethers, 2000), and there is no example of extensive uptake of LAP based applications in industry (Dumay, Dietz, & Mulder, 2005). Thus, the LAP movement has been confined to academic scholars, and is not widely known nor practiced outside the narrow borders of the LAP community (Lyytinen, 2004).

### 2.2.3 Criticisms on LAP approach

While the previous section showcases diverse applications to demonstrate the wide variety of interest in the LAP approach, LAP has also received some criticisms. Most of these criticisms are aimed specifically towards the use of speech act theory, but some focus on the general use of the LAP approach (Schoop, 2001). These criticisms are presented in this section along with solutions intended to counter them.
Bowers (Bowers, 1992) provides a critical attack on the LAP approach, specifically on the Coordinator, that heavily relies on formal speech act based representation for expressing networks of communication structures. He attacks the notion of formal representations, that claims to make a task easier and clearer (Schoop, 2001). He argues that the notion of formalism includes the possibility of forcing people to behave according to representations, thus, leading to situations of centralized power, where certain users can unduly influence others (Bowers, 1992). Therefore, Bowers claims that communication structures developed using the speech act theory oppress users because these structures enforce disciple and control (Schoop, 2001). Typically users are not aware of the vocabulary and intentions behind each speech act, hence, formalisms should be decentralized and kept open to critical assessment (Bowers, 1992). Schoop suggests that one strategy to achieve this goal is to use a participatory design to develop speech act based formalisms (Schoop, 2001).

Building on Bower arguments, Suchman provides the most challenging attack on the LAP approach (Suchman, 1994). She argues that LAP focuses on the speaker’s utterance and the hearer’s reaction rather than the interaction between the two, where the actual meaning and intentions emerge (Suchman, 1994). She further argues that predefined communication structures act as a plan with a set of actions to be executed, however, these actions are performed depending upon certain contexts and they are loosely connected to the plan (Suchman, 1994). She suggests that systems, instead of forcing predefined plans on to users, should keep track of actions and warn users of potential breakdowns (Suchman, 1994). Similar criticism of the LAP approach, in particular on speech act taxonomies, was provided by Ljungberg and Holm (Ljungberg & Holm, 1997) and suggested alternative criteria for classifying speech acts.
Allwood (Allwood, 1977) suggests that the classification of speech acts should be based on communicative functionalities and context (Schoop, 2001).

As a result of these criticisms on the usage and shortcomings of speech act theory, many later applications based on the LAP approach used Habermas’ theory of communicative action (Habermas, 1984) in addition to speech act theory (Schoop, 2001). Habermas criticizes the shortcomings of the speech act theory, while agreeing on its fundamental philosophy that communication is used to perform actions (Yetim & Bieber, 2003). Habermas developed his own version of speech act theory to provide a framework for understanding social interactions as coordinations of speech acts (Cecez-Kecmanovic & Webb, 2000). Habermas suggests that social interactions should be viewed from three dimensions (Goldkuhl, 2005; Habermas, 1984; Yetim & Bieber, 2003): (i) an ontology of three worlds (objective, subjective, and social); (ii) the pragmatics of language (representation, expressivity, and appellative); and (iii) the concept of validity claims (truth, normative rightness, sincerity, and comprehensibility). Habermas’ theory of communicative action provides a foundation for understanding and analyzing social interactions as the coordinations of communicative actions towards achieving mutually agreed upon goals (Klein & Huynh, 2004). This theory also provides a framework for identifying and addressing breakdowns in social interactions (Yetim, 2002). Like any other theory, the use of theory of communicative action in information systems was agreed upon by some researchers (Goldkuhl, 2000; Hirschheim, Klein, & Lyytinen, 1996; Klein & Huynh, 2004; Reijswoud, Mulder, & Dietz, 1999; Yetim, 2002) and criticized by others (Brooke, 2002; Doolin & Lowe, 2002; Introna, 1996; Ljungberg & Holm, 1997).
2.3 The Shifting Paradigm and its Issues

Over the past few years, the computing paradigm has been shifting from the current mainstream object-oriented perspective (Booch, 1993) to the service-oriented perspective (Papazoglou & Georgakopoulos, 2003). This type of change occurs when an existing paradigm reaches its limits to deal with increasing levels of software complexity. With the advent of Internet-based technologies, enterprises are redesigning their information systems to support their business activities over the Internet (Su, Lam, Lee, Bai, & Shen, 2001; Waldt & Drummond, 2005). Business applications that are implemented over heterogeneous internet technologies need to be loosely coupled, dynamically bound, and interoperable within and across organizations (Jain & Zhao, 2003). The object-oriented paradigm provides inadequate support for implementing applications over the Internet because object-oriented applications are rigid and hard to evolve (Henders, 1998). The advent of the SOC paradigm is intended, on the other hand, to facilitate business collaboration and application integration through open Internet standards (Papazoglou & Georgakopoulos, 2003). The SOC paradigm, is therefore considered the appropriate infrastructure for conducting seamless and automated business over Internet (Alonso et al., 2004).

Web service is the most promising technology based on the SOC paradigm, and therefore, it is the most promising technology for enterprise integration (Curbera, Nagy, & Weerawarana, 2001; Stal, 2002). A common goal of web service architecture and its associated standards is, thus, to support and coordinate communication among services to achieve particular business goals (Gottschalk et al., 2002; Papazoglou, 2003; Umapathy & Purao, 2007a). However, such a fundamental shift in the computing paradigm has produced new challenges and problems that
must be addressed by web services to support business activities over the Internet (Khalaf, Mukhi, & Weerawarana, 2003; Tsai, 2005). Some of the critical technical issues that web services must address are:

- What is a web service? What are its underlying architectures? (Aoyama et al., 2002)
- What are the set of key concerns—both functional and non-functional—that permit web services to support complex business activities? How to segregate these key concerns? (Hailpern & Tarr, 2001)
- What are the design methodologies and engineering principles for developing and deploying web services based applications? (Aoyama et al., 2002; Papazoglou & Yang, 2002)
- What are the appropriate models for web services that can support complex business interactions in systematic ways? (Curbera et al., 2001)
- How to integrate web service technologies into existing infrastructures? How to use web service technologies to integrate existing infrastructures? (Foster, Kesselman, Nick, & Tuecke, 2002; Hammer, 2001)

2.3.1 The LAP approach for web services

The LAP approach uses two key principles for understanding, designing, and developing information systems (Winograd, 2006). The first principle “language as action”, states that information exchange is not only used for sharing information but also to perform actions (Flores et al., 1988). The second principle is that communicative action is used as a basis for understanding and coordinating business activities to achieve business goals (Weigand & de Moor, 2003). Thus the LAP approach offers a framework for revealing underlying structures that
enable participants to act more effectively when efficient coordination is required (Winograd, 2006). With a communication oriented perspective similar to web services, the LAP approach is likely to be an appropriate framework for understanding, designing, and developing underlying web service architecture and associated standards because it assigns meaning to the activities performed through web services to achieve business goals.

The technical issues listed in the previous section portray the challenges created by shifts within the fundamental paradigms for developing business applications, from objects, or computational entities that contain data structures and methods (Booch, 1993), to services, or autonomous, platform-independent computational entities (Papazoglou et al., 2005). In order to be successful, web services must address these fundamental challenges. Some of the central concerns of web services are precisely those addressed by the LAP approach—segregating of key functionalities, coordinating communication among services to support complex business interactions, and developing communication-oriented design methodologies and principles. LAP, with its essential qualities of communication and coordination, is likely to be an appropriate candidate that web services can use to address its challenges. This dissertation is aimed at investigating that appropriateness.
CHAPTER 3 RESEARCH QUESTIONS AND APPROACH

Relevant publications

1. A Study of Language-Action Perspective as a Theoretical Framework for Web Services
   Published at: International Ph.D. Symposium on Service-Oriented Computing.

2. A Study of Language-Action Perspective as a Theoretical Framework for Web Services
   Published at: 2007 IEEE International Conference on Services Computing - Ph.D. Symposium Workshop (SCW 2007).
3.1 Research Questions

The fundamental hypothesis of this dissertation is that the Language-Action Perspective (LAP) would be an appropriate theoretical framework for web services. I test this hypothesis through three questions.

RQ 1. Can LAP constructs describe and explain the web services architecture?

RQ 2. How can LAP constructs be used for supporting the design of web services solutions?

RQ 3. Is design support with LAP constructs effective for designing web service solutions?

The three essays that form the core of this dissertation research are, therefore, driven by these three questions (see figure 3.1).

The first essay investigates the appropriateness of LAP to describe and explain web services architecture. It builds a reference framework based on LAP constructs, which identifies specific functions that need to be addressed in the interoperable web services architecture. The purpose of the framework is to provide guidance to all stakeholders of web services to make sense of its architecture.
The second essay demonstrates the appropriateness of LAP constructs for developing web service solutions. The domain selected to explore the second research question is enterprise integration (Hohpe & Woolf, 2004). The purpose of this essay is to build a design artifact that uses LAP constructs to assist in the design and development of conversations among web services for large-scale enterprise integration solutions.

The third essay evaluates appropriateness of LAP constructs embedded in the design artifact developed in the second essay. The purpose of this evaluation is to find empirical support for the relative effectiveness of the theoretical constructs to build the design artifact (Hevner, March, Park, & Ram, 2004).

### 3.2 Research Methodology

As this dissertation involves three inter-dependent yet distinct research essays, employment of multi-methodologies is highly appropriate due to its ability to produce richer and more reliable results (Mingers, 2001, 2003). The use of multiple methods allows a researcher to handle the limitations in any one method by building on the strengths of the other methods (Sawyer, Cooprider, Galliers, Gallivan, & Kaplan, 1997). Therefore this research will be conducted using multiple methods, each essay using different research methodologies.

#### 3.2.1 First Essay: Conceptual Development and Logical Arguments

The objective of the first essay is to develop a reference framework that describes and explains web service architecture. This reference framework is constructed based on my pre-understanding of Language-Action literature and web services technology. Based on this pre-
understanding, I conceptualize known information that is relevant to web services from Language-Action literatures and connect the gathered concepts through logical arguments to construct a framework. Therefore, the first essay uses conceptual development and logical arguments as its research method.

The methodology adopted in this essay, is similar to Weick’s approach to theory construction (Weick, 1989). According to Weick (Weick, 1989), theory construction is very similar to the concept of sensemaking. Sensemaking is the process through which tacit, private, complex, random knowledge is made explicit, public, simpler, ordered, and relevant to the situation at hand (Weick, Sutcliffe, & Obstfeld, 2005). I, therefore, use the sensemaking process to conceptualize the known information relevant to web services from Language-Action literature and logically connect those gathered concepts to construct a reference framework that provides guidance for the web service community’s efforts to make sense of its architecture.

3.2.2 Second Essay: Design Science Research Framework

The objective of the second essay is to construct a software artifact to assist integration designers with the development of integration solutions. The software would provide heuristic information for the selection and use of appropriate design knowledge to develop web service solutions. This heuristics information should be relevant and applicable (Brocke & Buddendick, 2006) and also should be specified and constructed using appropriate theoretical foundations (Weber, 2003). The design science research paradigm (Hevner et al., 2004) provides a framework to strengthen the construction of information technology artifacts that is theoretically and empirically
grounded (Brocke & Buddendick, 2006). Therefore, the second essay uses a design science research framework as its research method.

The framework of design science was established based on the influential book by Simon, *The Sciences of the Artificial* (Simon, 1996). In this seminal work, Simon identifies the need for various design sciences with the perspective of design as a problem solving activity primarily to create an innovative technological product (Venable, 2006). Simon work’s has influenced information systems (IS) scholars and they have adopted it to examine complex, artificial, and purposefully designed IS (Brocke & Buddendick, 2006; Carlsson, 2006). Based on the use of this approach in the IS field and the significance of the findings and relevance, a research framework for IS known as “design science research” (Hevner et al., 2004) was introduced (March & Smith, 1995).

The design science research framework provides seven guiding principles for conducting design science research (Hevner et al., 2004). The seven principles are (i) the research must produce an innovative and purposeful artifact; (ii) the artifact must address a specific organizational problem; (iii) the investigator must demonstrate the utility and quality of the artifact through rigorous evaluation; (iv) the investigator must provide verifiable research contributions that solve the intended problem or solves the problem in a more effective or efficient manner; (v) the investigator must apply rigorous methods for constructing and evaluating the artifact; (vi) the investigator must provide effective mechanisms for searching effective solutions for a given problem; and (vii) the investigator must communicate the research results with other researchers. These principles describe what constitutes good (i.e., rigorous and relevant) design science
research, therefore, design science researchers should address these principles in some manner (Carlsson, 2006; Hevner et al., 2004).

3.2.3 Third Essay: Controlled Experiment

The objective of the third essay is to evaluate effectiveness of the web service solutions developed using the software artifact built in the second essay. The evaluation attempts determine the cause and effect relationship between effective web service solutions and the software artifact (Tvedt & Collofello, 1995). Controlled experiment (Shadish, Cook, & Campbell, 2001) is the preferred strategy to determine such cause and effect relationships (Sjoeberg et al., 2005). Therefore, the third essay uses controlled experiment as its research method.

A controlled experiment, in the context of software engineering, is defined as an experiment in “which individuals or teams (the experimental units) conduct one or more software engineering tasks for the sake of comparing different populations, processes, methods, techniques, languages, or tools (the treatments)” (Sjøberg et al., 2005). Controlled experiments, typically, include treatment and control or comparison groups, ideally with the randomized assignment of subjects to treatment groups (Kuehl, 2000). Such experimental studies attempt to establish cause and effect relationships between a particular treatment of interest and some desirable outcome, with the potential for statistically significant results (Basili, 2006). These relationships are established by isolating treatment effects from other possible determiners of outcomes through the use of control comparison groups, so that one group is given the treatment of interest and another group is not provided the treatment (Kuehl, 2000).
Relevant publications

1. A Theoretical Investigation of the Emerging Standards for Web Services  
   *Co-authored with*: Sandeep Purao  
   *Published at*: Information Systems Frontiers, vol 9(1), Pg.: 119-134.  

   *Co-authored with*: Sandeep Purao  
   *Published at*: the International Working Conference on the Language-Action Perspective on Communication Modelling (LAP), 2004, Pg.: 41-58  
   *URL*: [http://www.scils.rutgers.edu/~aakhus/lap/Purao.pdf](http://www.scils.rutgers.edu/~aakhus/lap/Purao.pdf)
4.1 Introduction

Standards are an important component for web services because they facilitate interactions among applications within and across organizations. They must, therefore, cover a wide array of concerns such as messaging, publishing, discovering, and composing. Constructing a single, monolithic standard that encompasses all these is enormously difficult. Ongoing work on web services standards\(^1\) has, therefore, developed separate core standards for publishing (WSDL (WSDL, 2001)), finding (UDDI (UDDI, 2004)), and binding (SOAP (SOAP, 2003)). Such ‘separation of concerns’ is a necessary attribute of web services standards, which can be achieved by segmenting the standards-space into different ‘layers’ (similar to (Kreger, 2001)). The resulting ‘standards stack’\(^2\) segregates the standards space into multiple layers which can provide boundaries for development and use of standards. An analogy can be seen in the OSI seven-layer model (Bodhuin, Esposito, Pacelli, & Tortorella, 2004), which separates networking technologies into a seven-layer abstraction. A standards stack for web services can perform a similar function.

An important concern, however, is the identification of appropriate layers in this stack. Without an agreement about these layers, development of standards in this space remains a piecemeal effort. For example, standards that cross layers, such as WS-BPEL (WS-BPEL, 2005) and WS-CDL (WS-CDL, 2005), have been proposed over the last few years. Competing standards within a layer have also been proposed, such as WS-coordination (WS-CF, 2005) and WS-CDL (WS-CDL, 2005), causing confusion for information systems developers.

\(^{1}\) Appendix A shows a list of standards proposed for web services following different initiatives.

\(^{2}\) The term ‘web services stack’ refers to a ‘stack of web service standards’ (Sleeper & Robins, 2001).
Our review of prior work shows that holistic efforts to identify these layers have been lacking. Prior work has either produced compilations of existing standards (Mukhi et al., 2004) or has been driven by considerations of adoption of standards (Gosain, 2003). Without the benefit of a holistic framework guided by appropriate theories, work in this space has suffered from false starts (WS Arch, 2005; WSCI, 2002).

An appropriate theoretical framework is found in the stream of research known as Language-Action Perspective (LAP), which maps well to the core of service-oriented computing, i.e., ‘communication’ (Lemniotes, Papadopoulos, & Arbab, 2004). LAP was originally developed in the context of human actors communicating with one another to achieve organizational goals (Lyytinen, 2004). We argue that this research stream can be adapted and applied to the web services standards space. Our objective in this paper is to adapt research from the Language-Action perspective to construct a reference framework for the web services standards space, and demonstrate how it can be used to assess existing standards or develop new standards. The paper starts by investigating how the three existing initiatives operationalize core web services concepts. Next, by adapting important concepts from LAP, we develop a reference framework for the web services standards space. We then demonstrate how the reference framework can be used to assess the existing initiatives. Finally, we discuss implications for refinement and development of future standards.
4.2 Web services as the realization of service-oriented computing

The service-oriented computing (SOC) paradigm\(^3\) is currently being realized through three different initiatives. The first represents a major effort from the World-Wide Web Consortium (W3C), which builds on the premise that web services may be defined in a programmatic manner so that companies can use them to integrate their operations (WS Arch, 2005). The second represents an effort that is backed by the research community interested in a vision of the Semantic Web that augments web services with semantic components (Paolucci & Sycara, 2004). The third represents an effort by the Organization for the Advancement of Structured Information Standards (OASIS) in conjunction with the UN as a way to build upon existing Electronic Document Interchange (EDI) infrastructure to facilitate global trade (ebXML-Req, 2001).

4.2.1 Alternative instantiations

The three initiatives conform to the same basic operations (publish, find and bind)\(^4\) and roles (service provider, service discovery agency, and service requestor) (Manes, 2003; Papazoglou & Georgakopoulos, 2003). Each, however, operationalizes these with slight differences. We explain and contrast the three with an online travel agent example:

“[WSClient], a potential customer, queries a business registry for online travel agent. The registry returns a list of online travel agent services. [WSClient] selects [TAService] service, which is most fitting to its requirements and then binds to that service.”

---

\(^3\) Service-Oriented Computing (SOC) refers to a re-conceptualization of software to its essence, that of service (Papazoglou & Georgakopoulos, 2003; Turner, Budgen, & Brereton, 2003). In an SOC environment, applications are recast as “services.” They declare their functional and nonfunctional requirements and capabilities in an agreed upon, machine-readable format (Curbera et al., 2003).

\(^4\) A native capability of SOC applications is the ability to describe themselves (publish), locate service partners (find), and invoke these services as required (bind). These three operations provide the basic building blocks of a service-oriented architecture (Curbera et al., 2003).
4.2.1.1 The W3C initiative

To realize our example scenario using the W3C initiative, [TAService] would create a Web Services Definition Language document (WSDL, 2001) to describe its service interfaces, and publish it in the Universal Description, Discovery, and Integration registry (UDDI, 2004). [WSClient] will query the registry for services, which provide online travel agent capabilities. [WSClient] would select a service which meets its requirements. Assuming that [WSClient] selects [TAService], it would then bind its application to [TAService]. [WSClient] will generate Simple Object Access Protocol (SOAP, 2003) messages conforming to [TAService]’s service declarations, and invoke [TAService]. Both [WSClient] and [TAService] will now exchange messages to communicate.

4.2.1.2 The semantic web services initiative

The key difference between the W3C initiative and the semantic web services initiative is that the first depends on a syntactic description of web services, whereas the second utilizes more semantic descriptors derived from the OWL-based Web Services Ontology (Ankolekar et al., 2001). To realize our example scenario using Semantic Web Services, [TAService] would create a service profile of its capabilities using the semantic descriptors. The service profile contains a service model that describes how to interact with the service, and a service grounding that maps the information exchanges described in the service model into actual messages (Ankolekar et al., 2001; Paolucci, Sycara, Nishimura, & Srinivasan, 2003). The service profile is then published in a Service Registry. [WSClient] will query the Service Registry to find a required service, and when found, use its service grounding to bind the selected service. Assuming that [WSClient] selects [TAService], both services can then generate messages to communicate.
4.2.1.3 The ebXML initiative

Unlike the first two, the ebXML initiative builds on existing EDI standards (ebXML, 2005) to specify the ebusiness XML language that globally distributed business partners can use to signify their compliance with minimum requirements for trading and conducting business (ebXML, 2005). The example scenario is realized following the ebXML initiative in the following manner. [TAService] would request the Business Process Specification Schema (BPSS) (ebBPSS, 2001) from an ebXML registry (ebRS, 2002) and populate it with its own capabilities that describe its implementation of an online travel agent service along with a Collaboration Protocol Profile (CPP) (ebXML-CPPA, 2002) that specifies the electronic interactions it can participate in. [TAService] will then submit the BPSS and CPP, i.e., its business profile to the ebXML registry. When [WSClient]’s query returns [TAService] as a potential business partner, it can download [TAService]’s business profile from ebXML registry. Both [TAService] and [WSClient] may then agree to conduct business (using their CPP), and will produce a Collaboration Protocol Agreement (CPA). Once the CPA is in place, [TAService] and [WSClient] are said to possess the required trading partner information, and may engage in conducting business electronically using a messaging service that is part of the ebXML specification (Rawlins, 2002).

4.2.2 Commonalities across instantiations

The three initiatives are similar, yet different from one another in their vision because of the different challenges they see at the core of the web services paradigm. For W3C, the key challenge is providing a set of application programming interfaces (APIs) that will allow existing applications to communicate with each other over the web (WS Arch, 2005). For the semantic web services research community, the challenge lies in describing and discovering web services
not only syntactically but also semantically (Paolucci & Sycara, 2004). The third, OASIS, views the core challenge as developing a framework that utilizes existing EDI infrastructure. It, therefore, provides a consistent and uniform manner for exchange of electronic business data for B2B and B2C environments (ebXML-Req, 2001). Over the years, these three initiatives, which started as separate endeavors, have interacted with one another further emphasizing their commonalities. Figure 4.1 highlights the commonalities among the three initiatives.

![Figure 4.1. Comparing alternative instantiations with the travel agent scenario](image)

The figure shows that ‘communication’ is at the core of the service-oriented computing paradigm (Lemniotes et al., 2004) – it is used to publish, find and bind services. The mechanism used to accomplish these actions is communication (e.g., publishing a service in a UDDI

---

5 For example, the W3C has a working group on semantic web services (WS SWSIG, 2002). The UDDI standard, initially developed by OASIS (OASIS-CPPA, 2002), is now part of the core W3C standards, and the ebXML standard advocated by OASIS (ebXML, 2005) is being integrated within the W3C efforts (WS Activity, 2005).
corresponds to the *action* of ‘advertising available capabilities’). Communication, thus, represents ‘action-taking’ in the realm of web services and includes actions such as: (a) advertising available capabilities, (b) locating partners, (c) establishing commitment, (d) negotiating contract terms, (e) entering into a contract, (f) carrying out a transaction, (g) performing an exchange, (h) carrying out processes, (i) establishing trust, and (j) establishing relationships. These activities closely correspond to *business* activities that increase in time span (from facilitating single interactions with business partners to carrying out processes that include multiple interactions to facilitate business relationships that may include multiple processes). This view of communication and language is different from the traditional database perspective, which views language as a mechanism to record facts from the universe of discourse (Brodie, Mylopoulos, & Schmidt, 1984; Elmasri & Navathe, 2003). The next section builds on this perspective to identify a theoretical perspective for the web services standards space.

### 4.3 A theoretical framework for the web services standards space

The two arguments above, (a) communication as action, and (b) emphasis on business activities, lead us to suggest Language-Action Perspective (LAP) as an appropriate theoretical perspective for the web services standards space. Following LAP, we can view information systems as actors, i.e., communicative social entities, with different roles, knowledge, and processes (Klein & Huynh, 2004), engaged in performing business activities A number of frameworks (e.g., SAMPO (Auramaki et al., 1988) and DEMO (Dietz, 1994)) have been constructed on this foundation for modeling of *business* activities as *communicative* actions.
The choice of LAP as a theoretical perspective also meets the criteria for theory selection suggested by Holmstrom and Truex (Holmstrom & Truex, 2001): (a) selected theory’s historical context, (b) selected theory’s sensitivity towards details of the phenomenon under study, (c) selected theory’s impact on the choice of research method and (d) selected theory’s contribution to cumulative theory-building.

The first criterion is met by the historical application of LAP to message-oriented phenomena. Other information systems based on LAP theories include Coordinator (Winograd & Flores, 1986), SAMPO (Auramaki et al., 1988), Action Workflow (Medina-Mora et al., 1992), DEMO (Dietz, 1994), MILANO (De Michelis & Grasso, 1994), and BAT (Goldkuhl, 1996). These systems provide a historical lineage that makes LAP theories appropriate for application to the new domain of web services. The second criterion ensures that the selected perspective focuses on appropriate constructs. LAP theories do, indeed, focus on communication instead of data (Verharen et al., 1996) following the constructs suggested by theories of speech acts, hermeneutics and sociology of knowledge (Goldkuhl & Lyytinen, 1982). The third criterion, which describes how the selected theory impacts the choice of research method, is largely inapplicable in the context of this work because our research method relies on the conceptualization of new reference architecture, and an analysis of existing standards as a validation mechanism. The final criterion assesses whether the selected theoretical perspective would contribute to cumulative theory-building in the target domain. LAP views an organization as a network of interrelated conversations (Aakhus, 2004) similar to actions performed by web services in concert with others. This perspective has been largely absent in web services. By
suggesting this perspective, our selection directly contributes to theory-building in the target
domain of web services standards.

4.3.1 A brief review of research on the language-action perspective

The use of Language-Action perspective (LAP) for information systems can be traced to Flores
and Ludlow (Flores & Ludlow, 1980), who argued that human beings are linguistic beings and
act through language (Schoop, 2001). LAP formulates a norm-based and interpretive alternative
of how language is constituted in social life to analyze its implications for the design of
information systems (Auramaki et al., 1988; Goldkuhl & Lyytinen, 1982, 1984; Hirschheim et
al., 1995; Lyytinen, 2004). It conceives of the use of information systems primarily as a medium
or tool for communication (Goldkuhl & Lyytinen, 1982) drawing upon linguistic and social rules
that govern the use of language (Goldkuhl & Lyytinen, 1984). It is, therefore, not so much
centered on perfecting computational models and techniques. Instead, it seeks to explain and
understand relations between computational phenomena and social behaviors that are embedded
in or enabled by information systems (Lyytinen, 2004). The Language-Action approach is, thus,
based on the premise that much in organizations is performed through language, i.e.,
communication is primarily action which, in turn, facilitates coordination and interaction
(Ljungberg & Holm, 1997).

An action view on language and communication for the analysis of business activities is, thus,
the essence of LAP. An important theoretical foundation for LAP is the Speech Act Theory
(Austin, 1962; Habermas, 1984; Searle, 1969). Uttering a sentence is the performance of a
purposeful act, a so-called speech act. LAP emphasizes the need for regarding the performance
of business processes as patterns of inter-related speech acts (Flores & Ludlow, 1980). Winograd & Flores (Winograd & Flores, 1986) extend this idea and introduce a conversation-for-action schema, which covers a number of state changes, e.g., someone (A) states a request, someone else (B) makes a promise and then reports completion, which in the end A declares completed. Several frameworks have been developed based on the language-action perspective, such as a layered pattern approach for electronic commerce transaction (Weigand et al., 1998), generic layered patterns for business modeling (Lind & Goldkuhl, 2001) and an aggregative atoms, molecules and matter model (Dietz, 2002). Other approaches include Action Workflow (Medina-Mora et al., 1992), Dynamic Essential Modelling (DEMO) (Dietz, 1994, 2001; Reijswoud et al., 1999), and Business Action Theory (BAT) (Goldkuhl, 1996, 1998; Lind & Goldkuhl, 1997, 2001). These efforts, which have built a layered view of communication-oriented business activities, provide further support to the argument that LAP can provide the basis for understanding the web services standards space.

4.3.2 An LAP-inspired framework for the web services standards space

Developing an LAP-inspired framework for the web services standards space requires one key adaptation of the premises underlying LAP. The participants in the communicative action represent computationally described web services instead of organizational actors (Aakhus, 2004). As established in the previous section, the objective remains carrying out specific business activities. Following LAP, this can be achieved if (a) there is a common platform through which participants can share, exchange and reach common understanding on assumptions, values and norms of their interaction; (b) participants can advertise, negotiate and reach mutual commitments for performing purposive business activities; and (c) participants
engage in rational discourse to perform their committed business actions and resolve any breakdowns that may occur during their interaction (Klein & Huynh, 2004). The framework we propose, accordingly, consists of three levels: (a) **communication platform**, (b) **communicative act**, and (c) **rational discourse**.

### 4.3.2.1 Communication platform

The first level is the enabler of communicative acts between communicating parties. In LAP, the process of performing a communicative act is a social action (Goldkuhl & Ågerfalk, 2000; Searle, 1969). In order to make a communicative act successful, four conditions are required (Dietz, 2001, 2004):

- There must be a communication channel, i.e., there should be a transportation protocol to transport communicative messages between parties.
- There must be a common syntax, so that messages are correctly recognized by the parties.
- There must be common semantics, so that messages are correctly interpreted by the parties.
- There must be common social culture between parties, so that they have full agreement on the commitments raised by the communicative acts.

The above conditions suggest that a prerequisite to successful communicative acts is a communication platform. It can be divided into three layers: [channel], [messaging] and [guarantee]. The first condition establishes the need for a communication channel between parties as a conduit of messages. The second and third conditions establish the need for common syntax and semantics for messages. The fourth condition guarantees that the expectations for interpretation of messages are shared by the communicating parties.
4.3.2.2 Communicative act

The second level creates commitments between communicating parties (Dietz, 2001) for some future action (Searle, 1969) and creates a “shared understanding” against a shared background (Habermas, 1984). The participants achieve this goal through four phases (Goldkuhl, 1996, 1998):

- First, the participant who has ability (capacity and know-how) to perform an action offers and exposes it in a form searchable by other participants.
- Second, a participant who needs certain performance of an action, searches for partners who offer this ability.
- Third, after the participants find each other, they establish contact, exchange proposals and negotiate with each other to reach an agreement (Schoop, 2002).
- Finally, the participants establish a formal contract which expresses mutual commitments of the participants for future actions.

The four phases directly map to four layers: [capability exposure], [capability search], [proposal and negotiation] and [contract establishment].

4.3.2.3 Rational discourse

The third level emerges out of the performance of patterns of communicative acts (Auramaki et al., 1988; Klein & Huynh, 2004; Lyytinen et al., 1987). Rational discourse occurs among actors, when they pursue their own goals and harmonize their plans of action based on a common definition of the situation (Cecez-Kecmanovic & Janson, 1999; Reijswoud et al., 1999). This coordination via rational discourse can include four layers:

- First, interaction between actors consist of patterns of triggers and responses (Dietz, 2002; Lind & Goldkuhl, 2001), i.e., conversations among actors are made up of a finite sequence of communicative acts (Dietz, 2002; Weigand et al., 1998).
- Second, a composition of related exchanges can lead to a goal state that is satisfactory to the needs of the actors (Dietz, 2002; Lind & Goldkuhl, 2001). Each exchange, thus, represents a
step that facilitates transition from an initial state to a goal state (Klein & Huynh, 2004; Lind & Goldkuhl, 2001).

- Third, recurring or long-term transactions require establishing relationships among actors, and sustaining or improving them over time (Lind & Goldkuhl, 2001).
- Finally, concurrent contracts may be run with several parties (Weigand et al., 1998). This includes the need to regulate long term contracts, and the ability to change business transactions as the contracts changes (Goldkuhl & Melin, 2001), i.e., it requires a global overview of running contracts and explicit control of the running transactions and existing relationships (Weigand et al., 1998).

These map to the four layers: [exchange], [transaction], [relationship management] and [managing concurrent contracts] respectively.

Table 4.1 outlines the LAP-inspired framework. The first level (at the bottom of the table), communication platform, describes ‘how’ communicative acts are enabled; the second level (in the middle of the table), communicative act, focuses on ‘what’ the acts themselves constitute; and the third level (at the top of the table), rational discourse, emphasizes ‘why’ patterns of communicative acts are carried out, e.g., to achieve business goals.

The next section demonstrates how the proposed ‘reference framework’ can be used to assess existing web services standards stacks.
Table 4.1  An LAP-inspired framework for the web services standards space

<table>
<thead>
<tr>
<th>Rational discourse (coordination among actors based on commitments for achieving business goals)</th>
<th>Managing concurrent contracts, i.e., managing contracts with multiple partners, “interactions between several contracts that run concurrent between several parties.” (Weigand et al., 1998), “a global overview, a representation of the contracts” (Weigand et al., 1998).</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Managing concurrent contracts] (Weigand et al., 1998)</td>
<td>Managing concurrent contracts, i.e., managing contracts with multiple partners, “interactions between several contracts that run concurrent between several parties.” (Weigand et al., 1998), “a global overview, a representation of the contracts” (Weigand et al., 1998).</td>
</tr>
<tr>
<td>[Relationship management] (similar to transaction group (Lind &amp; Goldkuhl, 2001), contract (Weigand et al., 1998))</td>
<td>Relationship management deals with multiple or recurring transactions with one partner following a long term contract, “The recurrent business transactions need to be framed within a wider agreement...” (Lind &amp; Goldkuhl, 2001), “…creation and sustainment of business relations” (Weigand et al., 1998).</td>
</tr>
<tr>
<td>[Transaction] (similar to business transactions (Lind &amp; Goldkuhl, 2001), business process (Dietz, 2002), workflow (Weigand et al., 1998), contract (Weigand et al., 1998))</td>
<td>A pattern built from different types of exchanges related to each other (composition of interrelated activities or interactions to achieve a goal) “...composition of connected interactions.” (Dietz, 2002), “…transaction includes a number of different exchanges, where each of these exchanges constitutes the ...transaction’s different phases.” (Lind &amp; Goldkuhl, 2001), “…a set of related transactions aimed at some goal.” (Weigand et al., 1998).</td>
</tr>
<tr>
<td>[Exchange] (Exchange in (Lind &amp; Goldkuhl, 2001), includes Action pair (Lind &amp; Goldkuhl, 2001), similar to interaction (Dietz, 2002), similar to transaction (Weigand et al., 1998))</td>
<td>A exchange involves interaction between actors consisting of patterns of trigger and response, “patterns of triggers and responses.” (Lind &amp; Goldkuhl, 2001), “An exchange means that one actor gives something in return for something given by another actor.” (Lind &amp; Goldkuhl, 2001), “…acts appear to occur in particular patterns. …..we call these patterns conversations. (Dietz, 2002), “the smallest possible sequence of actions (speech acts) that leads to a certain...state” (Weigand et al., 1998).</td>
</tr>
<tr>
<td>Communicative act (creating commitments between communicating parties)</td>
<td>Provider and customer enter into commitments to perform a business transaction. “Customer and supplier come to an agreement concerning the business transaction. The contract is a mutual communicative action expressing the mutual commitments made; i.e., commitments for future actions.” (Goldkuhl, 1998)</td>
</tr>
<tr>
<td>[Contract establishment] (Goldkuhl, 1996, 1998; Habermas, 1984; Lind &amp; Goldkuhl, 1997)</td>
<td>Provider and customer enter into commitments to perform a business transaction. “Customer and supplier come to an agreement concerning the business transaction. The contract is a mutual communicative action expressing the mutual commitments made; i.e., commitments for future actions.” (Goldkuhl, 1998)</td>
</tr>
<tr>
<td>[Proposal and negotiation] (Goldkuhl, 1996, 1998; Habermas, 1984; Schoop, 2002)</td>
<td>Participants negotiate with each other through interpretation of the situation at hand and seek to achieve consensus. “Bids and counter-bids are made. The desire and demand of the customer are expressed. The supplier can make different offers.” (Goldkuhl, 1998), “…characterized by highly dynamic and interactive exchanges that can range from simple orders from a product catalogue to complex negotiations involving offers, counter-offers, bargaining etc.” (Schoop, 2002).</td>
</tr>
<tr>
<td>[Capability search] (Goldkuhl, 1996, 1998; Habermas, 1984; Lind &amp; Goldkuhl, 1997)</td>
<td>Customers desiring a capability seek contact with providers providing the capability. “The customer does not have the corresponding ability … needs which may be satisfied by potential suppliers and their products (goods/services)” (Goldkuhl, 1998).</td>
</tr>
<tr>
<td>[Capability exposure] (Goldkuhl, 1996, 1998; Habermas, 1984; Lind &amp; Goldkuhl, 1997; V. E. v. Reijswoud &amp; M. Lind, 1998)</td>
<td>Participants offer and exposes their business in a form searchable by other participants, “ability is offered and exposed to the market” (V. v. Reijswoud &amp; M. Lind, 1998), “The supplier must have an ability (a capacity and a know-how) to perform business; to make offers and contracts and to fulfill these contracts.” (Goldkuhl, 1998)</td>
</tr>
<tr>
<td>Communication platform (preconditions to make communicative acts successful)</td>
<td>Ensuring delivery without distortion, “free from any kind of distortion, any form of coercion and ideology &quot;that excludes all force...except the force of the better argument.&quot; (Habermas, 1984).</td>
</tr>
<tr>
<td>[Guarantee] (Dietz, 2001; Habermas, 1984)</td>
<td>Ensuring delivery without distortion, “free from any kind of distortion, any form of coercion and ideology &quot;that excludes all force...except the force of the better argument.&quot; (Habermas, 1984).</td>
</tr>
<tr>
<td>[Messaging] (Dietz, 2001; Habermas, 1984)</td>
<td>Common syntax and semantics , “The concept of communicative action presupposes the use of language as a medium for a kind of reaching understanding...” (Habermas, 1984)</td>
</tr>
<tr>
<td>[Channel] (Dietz, 2001)</td>
<td>Conduit to carry messages</td>
</tr>
</tbody>
</table>
4.4 Assessment of existing web services standards stacks

4.4.1 Assessment of the W3C initiative

Table 4.2 summarizes the results of assessing the W3C initiative (WS Arch, 2005) against the LAP-inspired reference framework.

<table>
<thead>
<tr>
<th>LAP-inspired framework</th>
<th>Corresponding layer in W3C initiative (WS Arch, 2005)</th>
<th>Standards proposed (see Appendix A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rational discourse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Managing concurrent contracts]</td>
<td>Management</td>
<td>MUWS and MOWS</td>
</tr>
<tr>
<td>[Relationship management]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>[Transaction]</td>
<td>Aggregation</td>
<td>WS-BPEL</td>
</tr>
<tr>
<td>[Exchange]</td>
<td>Choreography</td>
<td>WS-CDL, WS-CF, WS-TXM</td>
</tr>
<tr>
<td>Communicative act</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Contract establishment]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>[Proposal and negotiation]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>[Capability search]</td>
<td>Discovery</td>
<td>UDDI</td>
</tr>
<tr>
<td>[Capability exposure]</td>
<td>Description</td>
<td>WSDL</td>
</tr>
<tr>
<td>Communication platform</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Messaging]</td>
<td>Messaging</td>
<td>SOAP</td>
</tr>
<tr>
<td>[Channel]</td>
<td>Communications</td>
<td>HTTP, FTP</td>
</tr>
</tbody>
</table>

The communication platform layers demonstrate the clearest mapping between the LAP-inspired framework and the W3C initiative. The communications layer outlines the requirements for communication between services similar to the [channel] layer in the LAP-inspired framework. Key standards in this layer include HTTP (HTTP-Webopedia, 2005) and FTP (FTP-Webopedia, 2005). The messaging layer facilitates communication between services by providing them a flexible and extensible mechanism for exchanging messages (Gottschalk et al., 2002; Kreger, 2003). It specifies a common syntax necessary for formulating messages. This layer, thus, corresponds to the [messaging] layer in the LAP-inspired framework. Messaging extensions
include security, addressing and reliability, which correspond to the notion of providing guarantees that ensure communication without distortions. This layer, thus, corresponds to the [guarantee] layer in the LAP-inspired framework. Some of the available standards in the Messaging and Messaging extensions layers include SOAP (SOAP, 2003) and extensions such as WS-Security (WS-Security, 2004), WS-Addressing (WS-Addressing, 2005) and WS-Reliability (WS-Reliability, 2004).

The communicative act layers demonstrate the least mapping between the LAP-inspired framework and the W3C initiative. The description layer provides a functional description of a service in terms of its interface and implementation (Turner et al., 2003); (Kreger, 2003). This layer, thus, corresponds to the [capability exposure] layer in the LAP-inspired framework. The key standard in this layer is the WSDL (WSDL, 2001). The discovery layer specifies mechanisms for discovering a service, i.e., locating a machine-readable description of a service (WS Arch, 2005). This layer, thus, allows service providers to publish their services descriptions to a business registry, so that service requestors can search and discover services that meet their requirements. This layer, thus, corresponds to the [capability search] layer in the LAP-inspired framework. The standard available in this layer is UDDI (UDDI, 2004). Layers from the LAP-inspired framework with no corresponding efforts in the W3C initiative, therefore, include [proposal and negotiation], and [contract establishment].

Finally, the rational discourse layers also demonstrate significant mapping between the LAP-inspired framework and the W3C initiative. The choreography layer defines the sequence and conditions under which messages are exchanged among services in order to achieve a goal state
(WS Arch, 2005). This layer, thus, corresponds to the [exchange] layer in the LAP-inspired framework. Available standards in this layer include WS-CDL (WS-CDL, 2005), WS-CF (WS-CF, 2005), and WS-TXM (WS-TXM, 2003). Second, the notion of aggregation in the stack proposed by the W3C initiative refers to the composition of web services (including any interaction and message flows between them (WS Arch, 2005)) and their representation as a higher-level web service (Gottschalk et al., 2002). This directly corresponds to the [transaction] layer in the LAP-inspired framework. Existing standards in this layer include WS-BPEL (WS-BPEL, 2005). Finally, the management layer provides a mechanism to monitor and control the execution of web services. It defines the model for managing web services as a resource. This corresponds to the [managing concurrent contracts] layer in the LAP-inspired framework. Standards in this layer include MUWS (WSDM, 2005) and MOWS (WSDM, 2005). Layers from the LAP-inspired framework with no corresponding efforts in the W3C initiative, therefore, include [relationship management]. The next two sub-sections assess the other two initiatives. The discussion that follows is not as comprehensive as that in section 4.1 to reduce redundancy.

### 4.4.2 Assessment of the semantic web services initiative

The semantic web services initiative provides a framework based on a set of roles and requirements for machine-readable semantic descriptions for deployment of web services (SWS Arch, 2005). Table 4.3 summarizes the results of assessing the semantic web services initiative against the LAP-inspired framework.
The **communication platform** layers demonstrate the clearest mapping between the LAP-inspired framework and the semantic web services initiative because they utilize the foundation provided by the W3C initiative, enhancing it with the Ontology Web Language (OWL, 2004).

### Table 4.3 Assessing the Semantic Web Services-initiative against the LAP-inspired framework

<table>
<thead>
<tr>
<th>LAP-inspired framework</th>
<th>Corresponding layer in the Semantic Web Services Initiative (SWS Arch, 2005)</th>
<th>Standards proposed (see Appendix A)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rational discourse</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Managing concurrent contracts]</td>
<td>Status monitoring; termination and compensation</td>
<td>SWSL-FOL</td>
</tr>
<tr>
<td>[Relationship management]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>[Transaction]</td>
<td>Contact initiation; status monitoring; termination and compensation</td>
<td>SWSL-FOL, OWL-S</td>
</tr>
<tr>
<td>[Exchange]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td><strong>Communicative act</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Contract establishment]</td>
<td>Service contract negotiation</td>
<td>SWSL-Rules</td>
</tr>
<tr>
<td>[Proposal and negotiation]</td>
<td>Client characterization of desired activity; Service contract negotiation</td>
<td></td>
</tr>
<tr>
<td>[Capability search]</td>
<td>Client characterization of service providers; service discovery process</td>
<td></td>
</tr>
<tr>
<td>[Capability exposure]</td>
<td>Published advertisement and service model</td>
<td>OWL-S, WSDL, SWSL-Rules</td>
</tr>
<tr>
<td><strong>Communication platform</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Messaging]</td>
<td></td>
<td>SOAP, OWL</td>
</tr>
<tr>
<td>[Channel]</td>
<td></td>
<td>HTTP, FTP</td>
</tr>
</tbody>
</table>

The **communicative act** layers also demonstrate significant mapping between the LAP-inspired framework and the semantic web services initiative. First, the *published advertisement and service model* layer provides protocols that service providers can use to advertise their services (SWS Arch, 2005) corresponding to the [capability exposure] layer in the LAP-inspired framework. Next, the *client characterization of service provider* sub-layer provides clients the ability to formulate abstract descriptions of requirements that candidate services must meet (SWS Arch, 2005). The *service discovery query process* sub-layer provides a protocol that a broker service can use to respond to client queries (SWS Arch, 2005). Together, these two sub-layers correspond to the [capability search] layer in the LAP-inspired framework. The next layer, **client**
characterization of desired activity, provides the means to exchange information about goals and capabilities of parties involved in the communicative act (SWS Arch, 2005). This layer corresponds to the [proposal and negotiation] layer in the LAP-inspired framework. The next layer, service contract negotiation, provides a protocol that service providers and clients can use to negotiate and establish contracts (SWS Arch, 2005). This layer corresponds to the [contract establishment] layer in the LAP-inspired framework. The standard that crosses all these layers is SWSL-Rules (SWSL, 2005).

The rational discourse layers demonstrate the least mapping between the LAP-inspired framework and the semantic web services initiative. First, the contract initiation sub-layer provides a protocol for clients to invoke the selected service. Next, the status monitoring sub-layer facilitates monitoring service execution, including mechanisms for stopping, resuming or restarting the service. The termination and compensation sub-layer provides a protocol for indicating success or failure. Together these three sub-layers correspond to the [transaction] layer in the LAP-inspired framework. The status monitoring and termination and compensation sub-layers also contain elements that correspond to the [managing concurrent contracts] layer in the LAP-inspired framework. Standards available for these layers include OWL-S and SWSL-FOL (SWSL, 2005). Layers from the LAP-inspired framework with no corresponding efforts in the semantic web services initiative, therefore, include [exchange] and [relationship management].

4.4.3 Assessment of the ebXML initiative

The ebXML initiative combines components from divergent XML initiatives to develop a single business standard (ebXML-Req, 2001) that can operate on existing EDI implementations
Table 4.4 summarizes the results of assessing the ebXML initiative against the LAP-inspired framework.

### Table 4.4 Assessing the ebXML-based initiative against the LAP-inspired framework

<table>
<thead>
<tr>
<th>LAP-inspired framework</th>
<th>Corresponding layer in the ebXML-based initiative (ebXML-TA, 2001)</th>
<th>Standards proposed (see Appendix A)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rational discourse</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Managing concurrent contracts]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>[Relationship management]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>[Transaction]</td>
<td>Business process and information modeling</td>
<td>BPIM</td>
</tr>
<tr>
<td>[Exchange]</td>
<td>Business process specification schema</td>
<td>BPSS</td>
</tr>
<tr>
<td><strong>Communicative act</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Contract establishment]</td>
<td>Collaborative protocol agreement</td>
<td>CPA</td>
</tr>
<tr>
<td>[Proposal and negotiation]</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>[Capability search]</td>
<td>Registry</td>
<td>ebXML RS, ebXML RIM</td>
</tr>
<tr>
<td>[Capability exposure]</td>
<td>Collaborative protocol profile</td>
<td>CPP</td>
</tr>
<tr>
<td><strong>Communication platform</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Guarantee]</td>
<td>Messaging service</td>
<td>SOAP, ebXML MS</td>
</tr>
<tr>
<td>[Messaging]</td>
<td>Communication protocol</td>
<td>HTTP, FTP</td>
</tr>
<tr>
<td>[Channel]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The **communication platform** layers show a clear mapping between the LAP-inspired framework and the ebXML initiative. First, the *communication protocol* layer utilizes standards such as HTTP and FTP for sending and receiving messages. This layer, therefore, corresponds to the [channel] layer in the LAP-inspired framework. Next, the *messaging service* layer provides the mechanism for sending ebXML messages over SOAP (ebMS, 2002). This layer, therefore, corresponds to the [Messaging] and [Guarantee] layers in the LAP-inspired framework.

The **communicative act** layers also show a significant mapping against the LAP-inspired framework. First, the *collaboration protocol profile* layer describes partner capabilities and service interface requirements (ebXML-TA, 2001). This layer contains CPP (ebXML-CPPA, 2002) as the key standard, and, therefore, corresponds to the [capability exposure] layer in the LAP-inspired framework. Second, the *registry* layer facilitates information sharing between
parties by providing a centralized repository (ebRS, 2002). It includes two elements: a registry service (interface for accessing the registry including interaction protocols, message definitions and schemas) (ebRS, 2002) and a registry information model (types of metadata of documents that are stored in the registry including relationships among the various metadata classes) (ebRIM, 2002). These elements correspond to the [capability search] layer in the LAP-inspired framework. The key standards in this layer include ebXML RS (ebRS, 2002) and ebXML RIM (ebRIM, 2002). Third, the collaboration protocol agreement layer describes the agreement between partners about their respective commitments and expectations (ebXML-CPPA, 2002; Ibbotson, 2001). The key standard in this layer is CPA (ebXML-CPPA, 2002). This layer corresponds to the [contract establishment] layer in the LAP-inspired framework. One layer from the LAP-inspired framework with no corresponding effort in the ebXML initiative, therefore, is [proposal and negotiation].

The rational discourse layers demonstrate the least mapping against the LAP-inspired framework. First, the business specification schema layer (ebBPSS, 2001) provides a nominal set of elements necessary to specify collaboration between business partners (Rawlins, 2002). This layer corresponds to the [exchange] layer in the LAP-inspired framework. The key standard in this layer is BPSS (ebBPSS, 2001). Next, the business process and information modeling layer provides mechanisms for trading parties to capture specific details of a business scenario and model a business process using a consistent modeling methodology (eBPOver, 2001). This layer corresponds to the [transaction] layer in the LAP-inspired framework. The key standard in this layer is BPIM (eBPOver, 2001). Layers from the LAP-inspired framework with no
corresponding efforts in the ebXML initiative, therefore, include [relationship management] and [managing concurrent contracts].

4.5 Discussion

The assessment of the three web services initiatives reported in the previous section suggests several recurring themes. As expected, the communication platform level appears to be common across the three standardization efforts, and shows the clearest mapping against the LAP-inspired framework. In particular, it shows clear agreement on the [channel] layer. This further translates to all three initiatives relying on standards recommended by the W3C. The agreement is also significant for the [messaging] layer (i.e., SOAP). The semantic web services initiative augment these basic standards by including OWL and the ebXML initiative augments these with additional syntax and delivery mechanisms. While no agreement has been reached yet on the [guarantee] layer (instantiated, for example, by the W3C initiative as WS-security and WS-reliability), adherence to common W3C standards on the lower layers suggests such agreement is a likely outcome in the near future.

At the communicative act level, all three initiatives provide standards for the [capability exposure] layer, although with slight variations. All three initiatives also provide standards for the [capability search] layer. The semantic web services initiative expands this to include the ability to specify desired characteristics of service providers. The semantic web services initiative is also the only initiative to provide standards for the [proposal and negotiation] layer. The W3C and ebXML initiatives indicate no standards for this layer, proposed or recommended. Useful directions for creating standards in this layer are available elsewhere (Jertila & Schoop, 2005; Schoop, 2003).
Finally, the W3C initiative is the only initiative that does not provide any standards for the [contract establishment] layer.

At the rational discourse level, the W3C initiative efforts are better developed than the semantic web services and ebXML initiatives. For the [exchange] layer, the W3C and ebXML initiatives provide competing, though well-developed alternatives. All three initiatives contain standards that provide partial support for the [transaction] layer. However, none provides protocols for conversation policies that can guide interaction between services including mechanisms to handle any exceptions. Useful research in this direction is available elsewhere, which can be leveraged for creation or refinement of standards (Fan, Umapathy, Yen, & Purao, 2004; Kimbrough & Yang, 2004; Moore, 2000, 2001; Umapathy, Purao, & Sugumaran, 2003).

None of the initiatives propose standards for managing relationships established through a contract or provide the ability to extend contracts over a long-term. As business transactions cross national and cultural boundaries, [relationship management] is an important aspect that can benefit from formalized approaches. This is a significant drawback for all three initiatives. Useful research in this direction is available elsewhere (Goldkuhl & Melin, 2001; Goldkuhl & Röstlinger, 1999). Finally, there is minimal support for the uppermost layer, [managing concurrent contracts]. The standards proposed by the W3C and semantic web services initiatives manage only a single contract or transactions related to that single contract, i.e., they do not provide a global overview of multiple contracts and their related multiple transactions as suggested by the LAP-inspired framework. Table 4.5 summarizes the above observations and provides pointers to additional research that may be useful for refinement of current standards.
Table 4.5  Assessment of the three standards initiatives against the LAP-inspired framework

<table>
<thead>
<tr>
<th>LAP-inspired Framework</th>
<th>Current Initiatives</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W3C</td>
<td>Semantic web</td>
</tr>
<tr>
<td>Rational discourse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Managing concurrent contracts]</td>
<td>Partial support</td>
<td>No support</td>
</tr>
<tr>
<td>[Relationship management]</td>
<td>No support</td>
<td></td>
</tr>
<tr>
<td>[Transaction]</td>
<td>Partial support</td>
<td></td>
</tr>
<tr>
<td>[Exchange]</td>
<td>Full support</td>
<td>No support</td>
</tr>
<tr>
<td>Communicative act</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Contract establishment]</td>
<td>No support</td>
<td>Full support</td>
</tr>
<tr>
<td>[Proposal and negotiation]</td>
<td>No support</td>
<td>Partial support</td>
</tr>
<tr>
<td>[Capability search]</td>
<td>Partial support</td>
<td>Full support</td>
</tr>
<tr>
<td>[Capability exposure]</td>
<td>Full support</td>
<td></td>
</tr>
<tr>
<td>Communication platform</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Guarantee]</td>
<td>Full support</td>
<td>No support</td>
</tr>
<tr>
<td>[Messaging]</td>
<td>Full support</td>
<td></td>
</tr>
<tr>
<td>[Channel]</td>
<td>Full support</td>
<td></td>
</tr>
</tbody>
</table>

One additional observation follows from the assessment of the three initiatives against the LAP-inspired framework. The framework suggests crisp boundaries built on the notion of “layers” as emphasized in prior work in the LAP research stream. This is apparent in the writings of Weigand (Weigand et al., 1998), Goldkuhl (Lind & Goldkuhl, 2001) and Dietz (Dietz, 2002) among others. Each has implicitly argued that the phenomenon (communicative action among actors in an organization) is complex, and needs to be unpacked into several layers to understand this complexity. This is particularly true for web services, where the standards should be carved into crisp components in order to ensure interoperability. Such crisp boundaries are not evident...
in many of the standards being developed negating the benefits of ‘separation of concerns’ known in the computing community. For example, the semantic web services initiative uses SWSL-Rules as a standard that spans as many as four layers (see Table 4.3). Similar outcomes are seen for the ebXML initiative, with standards spanning multiple layers in the communication platform (see Table 4.4). The W3C initiative appears to do well in this regard, perhaps due to the presence of multiple strong market participants, who may be implicitly enforcing the boundaries across layers as a way of ensuring separation of concerns.

We believe that the development and refinement of standards in a domain as important as web services should be guided by theoretical considerations. We hope that the analyses and assessment we have reported here, following the LAP theories, can suggest useful guidelines and constraints for these emerging standards. Our future work focuses on discovering foundational constructs that underlie LAP theories to provide insights into specific functionalities of web services standards.
CHAPTER 5 ACCESSING DESIGN KNOWLEDGE WITH SPEECH

ACTS TO GENERATE WEB SERVICE CONVERSATION SPECIFICATIONS

Research hypothesis: LAP is an appropriate theoretical framework for web services

Demonstrate Appropriateness

First Essay
Focus: Architecture Level

RQ1

Second Essay
Focus: Conceptual Level

RQ2

Third Essay
Focus: Empirical Level

RQ3

Relevant publications

1. Exploring Alternatives for Representing and Accessing Design Knowledge about Enterprise Integration
   Published at: 2007 Conceptual Modeling (ER) conference.

2. Assessment of Web Services Specifications to Support Long-Running Conversations
   Published at: Workshop on Information Technologies and Systems (WITS), 2006.

   Co-authored with: Sandeep Purao
   Published at: the International Conference on Conceptual Modeling (ER), 2006 (published as poster), Pg.: 586

4. Designing Enterprise Solutions with Web Services and Integration Patterns
   Co-authored with: Sandeep Purao
   Published at: the International Conference on Services Computing (SCC), 2006, Pg.: 111-118

5. Facilitating Conversations among Web Services as Speech-act based Discourses
   Co-authored with: Sandeep Purao and Vijayan Sugumaran
   Published at: Workshop on Information Technologies and Systems (WITS), 2003.
5.1 Introduction

The objective of this essay is to demonstrate appropriateness of Language-Action Perspective (LAP) constructs for developing web service solutions. This appropriateness is demonstrated for enterprise integration, a key application domain for web services. Enterprise integration (or system integration) refers to the capability of integrating disparate system functionalities (Jinyoul Lee, Siau, & Hong, 2003). Web services are an ideal implementation platform for integrating disparate systems because they are platform-independent (Iyer et al., 2003).

Business processes that reflect the requirements of integrated systems are prerequisite for the development of enterprise integration solutions (Gorton & Liu, 2005). Designing effective enterprise integration solutions also requires the appropriate structuring of the solution, for example, by utilizing appropriate design knowledge (Themistocleous, Irani, Kulj, & Love, 2004). An example of structuring would be the use of a Point-to-Point channel to ensure only one receiver will receive a particular message (Hohpe & Woolf, 2004). Finally, the designed solution must provide a path towards implementation such as web service specifications (Huang & Chung, 2003).

This research essay is, therefore, focused on these three activities with the goal of demonstrating the appropriateness of LAP for designing web service solutions. I develop LAP-based support for designers to access and select appropriate design knowledge, along with further support to convert their designed solutions into appropriate web service specifications. As proof of concept, the research outcomes are implemented in a research prototype following design science research methodology (Hevner et al., 2004).
5.2 Design knowledge

Design—the creation of a model, system, or artifact—is generally viewed as a complex, purposeful and goal-oriented activity (Gero, 1990). Simon (Simon, 1996) suggests that design is concerned with a “course of actions aimed at changing existing solutions into preferred ones” (pp 111). One definition of the design activity, therefore, consists a search process through the design alternatives to transform a set of requirements into a viable solution (Gero, 1990). In order to produce viable design solutions, designers require a broad range of skills, knowledge, and awareness (Friedman, 2000). Design activities involve the identification of issues and the exploration of various design strategies that may address those issues (Gero, 1990). Design strategies that can be employed to address specific design problems are known as design knowledge (Purcell & Sodersten, 2001). Thus, searching and applying appropriate design knowledge to address identified issues is a core design activity. In order to facilitate this search, designers need to access, acquire, and use design knowledge (Kalay, Swerdloff, & Majkowski, 1990). Consequently, making formulations and representations of design knowledge is an important concern.

A number of structures have been proposed to represent design knowledge, including: logic (expression through formal notations), rules (expression as IF condition THEN production rules), and structural forms (expression as conceptual structures with dependencies) (Jintae Lee, 1997; Salomons, van Houten, & Kals, 1993). In most forms, design knowledge consists of abstractions of design strategies and their associated relationships (Kalay et al., 1990). Depending on the design domain, these abstractions may consist of recurring patterns of geometrical, topological, temporal, causal, and functional relations among design knowledge (Goel, 1997). Explicitly
capturing design strategies in the form of patterns is known to aid in transferring design knowledge to other designers (Schmidt, 1995).

5.2.1 **Patterns as design knowledge**

The idea of patterns as a design solution was introduced by Christopher Alexander in the building construction architectures domain (Alexander, Ishikawa, & Silverstein, 1977), and first popularized in the software design domain by Gamma et al. (Gamma, Helm, Johnson, & Vlissides, 1994).

A pattern is a domain-independent abstraction of a common design structure for a recurring design problem. Each pattern describes when it can be applied, its design constraints, its consequences, and the trade-offs of applying it in particular contexts (Gamma et al., 1994). A pattern does not provide complex code; instead, it helps designers to describe and communicate design problems and solutions in a particular context.

When related patterns are woven together they form a “pattern language” which captures the relationship between solutions and problems (Hohpe & Woolf, 2004). Patterns and pattern languages guide designers through several decisions during software development. Patterns have been proposed for conceptual design (Coad, North, & Mayfield, 1995), in different domains for detailed design (Fowler, 1997; Gamma et al., 1994), and have been demonstrated to be viable for the conceptual design of new systems (Purao, 1998; Purao, Storey, & Han, 2003).
5.2.2 Enterprise integration patterns

For the domain of *Enterprise Integration*, a set of 65 patterns has been developed (Hohpe & Woolf, 2004). These patterns are abstract, thus, they do not provide implementation code or wrappers; instead, they provide recurring solutions that designers can use to solve integration problems. For instance, the request-reply pattern (shown in the figure 5.1) connects the requestor and replier applications via two channels: a request queue and a reply queue.

If a designer needs to connect two systems, where both systems need two-way communication, so that a requestor application sends a request message to replier application and replier application receives the message and sends a reply message to requestor application, then the designer can use this pattern.

![Request-Reply pattern](image)

**Figure 5.1. Request-Reply pattern**

The 65 Enterprise Integration Patterns (EIP) are organized into seven categories: integration styles, endpoint patterns, system management patterns, channel patterns, message construction patterns, routing patterns, and transformation patterns (Hohpe & Woolf, 2004). Respectively, the first three categories suggest different ways of exchanging documents, producing or consuming messages, and managing the performance of messaging systems; while the bottom four categories suggests different ways of integrating systems based on how they transport, construct, route, and transform messages. These last four categories are the core interest of this research.
5.3 Accessing enterprise integration patterns

To develop integration solutions, the designer needs to: (a) understand the integration requirements often represented in the form of business processes, and (b) identify the appropriate enterprise integration patterns based on this understanding (Gorton & Liu, 2005). For this, designers require analogical reasoning (Gentner, 1983), meaning that they must understand the mapping between a problem (requirements) and a solution (patterns) (Purao et al., 2003). This analogy reasoning requires knowledge of (1) the integration patterns, (2) the integration problem at hand, and (3) the constraints on the usage of patterns (Fernandez, 1998; Purao et al., 2003).

A key problem in facilitating this mapping is that there is a mismatch between these two perspectives. The integration requirements represent business activities and sequences of activities to be performed through a number of constructs such as sequencing, splits, joins, and iterations. Therefore, integration requirements follow the control-flow perspective (White, 2004; Wohed, Aalst, Dumas, Hofstede, & Russell, 2005). The integration patterns use constructs such as message, channel, one-way, and request-response that build abstract solutions to connect systems that perform business activities (Hohpe & Woolf, 2004). Therefore, integration patterns follow the operational perspective (Aalst, Hofstede, Kiepuszewski, & Barros, 2003).

To facilitate a mapping between these two perspectives, an appropriate mediator is necessary. The mediator should possess two prerequisites: (a) the ability to represent tasks represented in a business process specifications, and (b) the ability to represent messaging primitives in integration solutions. The latter (messaging primitives) may, then, be structured and concatenated to represent the former (tasks and task dependencies).
Speech acts are a viable candidate to play this role because they focus on the performance of actions necessary to accomplish tasks and the message exchanges that take place in aid of those actions (Johannesson & Perjons, 2001). Lim et al. (Lim, Juster, & Pennington, 1997) suggested that, enterprise integrations are represented by interactions among participants in a business process. Christiansson (Christiansson, 1998) suggests that interactions among participants are the basic unit for analysis and design of information systems that support business processes. Integration patterns contain integration tactics to support such interactions among participants (Hohpe & Woolf, 2004). These interactions can be operationalized with the use of structures of speech acts (Aakhus, 2004), because speech acts codify different actions that participants must perform to engage in these interactions (Bach & Harnish, 1979) (e.g. promises, orders, requests, and declarations).

5.3.1 Speech acts as mediator

Following the above arguments, we develop the mapping between business process specifications and integration patterns with interactions as the starting point. An interaction is defined as a sequence of interrelated speech acts performed by at least two performers (Aakhus, 2004), including an ‘initiator’ and a ‘responder’ (Christiansson, 1998). Speech act theory (Searle, 1969) argues that language can be used not only to describe a situation or fact, but also to perform action (Goldkuhl & Ågerfalk, 2000). In our case, speech acts represent the performance of actions by each participant in an interaction. Thus, sequences of speech acts codify the interaction. An important prerequisite for ensuring that appropriate speech acts are selected are action types that reflect the performers intent (Moore, 2001). Action types represent high-level
business actions performed by participants through communication directed towards other participant(s) (Lind & Goldkuhl, 2001).

To perform the mapping I developed several components. First, I represented each integration pattern as a sequence of speech acts. Second, I enhanced the representation of tasks in business processes with high-level action types. For the latter, I used the Business Process Modeling Notation (BPMN), a *de facto* standard that provides graphical notations for developing business process models (BPMN, 2006). Thus, the mapping between the BPMN notations and integration patterns consists of:

1. Identifying interactions within the business process specifications developed using BPMN notations.
2. Inferring the sequence of speech acts for each interaction.
3. Selecting enterprise integration patterns (EIP) based on the sequence of speech acts.

Interactions in the business process specifications can then be identified by detecting a sequence of tasks in the business process, where there are changes in the performer. The tasks carried out by one performer (initiator) are labeled as the first part of the interaction, and those carried out by the other performer (responder) are identified as the second part of interaction. Each task in the interaction can be mapped against a small set of possible speech acts using its associated action type. Arranged to follow the tasks, these interactions among speech acts can be mapped against representation of integration patterns as a sequence of speech acts. Figure 5.2 below provides overview of this mapping.
5.3.2 Components facilitating mapping

The components necessary for mapping described above include: (i) a parsimonious set of speech acts appropriate for enterprise integration patterns, (ii) a representation of enterprise integration patterns as a sequence of speech acts, (iii) a set of action types that depict high-level business actions and (iv) the associations between action types and speech acts.

5.3.2.1 A parsimonious set of speech acts

The initial set of speech acts that are relevant to the enterprise integration domain were obtained from pertinent literature such as Moore (Moore, 2001) and Johannesson and Perjons (Johannesson & Perjons, 2001). Moore (Moore, 2001) suggested that the speech act theory provides an appropriate foundation for automated electronic messaging systems and developed 23 speech acts that can aid in this automation. Johannesson and Perjons (Johannesson & Perjons, 2001) developed design principles for modeling business processes in the context of the enterprise integration domain and suggested 12 speech acts that can be used to represent different
message types that are typically found in the business processes. After removing repetitive speech acts in these two sets, they were combined to form an initial set of speech acts. This combined set was further refined by aggregating the speech acts that could encompass other speech acts into single speech acts. For instance, the *Query* speech act can encompass the *Request* speech act, as both represent the intent of the initiator querying for certain information from the responder. The refinement resulted in 9 speech acts: Acknowledge, Cancel, Commit, Direct, Disagree, Fulfill, Inform, Propose, and Query. Table 5.1 provides brief description of the parsimonious set of speech acts.

<table>
<thead>
<tr>
<th>Speech act</th>
<th>Description</th>
<th>Source</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledge</td>
<td>Acknowledges successful completion of an exchange or transaction</td>
<td>Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Cancel</td>
<td>Cancels an established commitment or contract</td>
<td>Johannesson and Perjons 2001 and Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Commit</td>
<td>Commits to a course of action or an obligation</td>
<td>Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Disagree</td>
<td>Disagrees on an issue with other participants</td>
<td>Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Direct</td>
<td>Attempts to get the recipient to perform a desired action</td>
<td>Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Fulfill</td>
<td>Completes certain material or economic actions</td>
<td>Johannesson and Perjons 2001 and Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Inform</td>
<td>Provides information that will be useful for other participants actions</td>
<td>Johannesson and Perjons 2001 and Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Propose</td>
<td>Proposes to create, change, or cancel a contract, commitment, or any other economic event</td>
<td>Johannesson and Perjons 2001 and Moore 2001</td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>Requests more information</td>
<td>Johannesson and Perjons 2001 and Moore 2001</td>
<td></td>
</tr>
</tbody>
</table>

These two speech acts were dropped later because they were unemployable in the context of integration patterns.
5.3.2.2 Representing integration patterns with speech acts

The parsimonious set of speech acts was used to identify a sequence of speech acts that can represent each integration pattern of interest. I acknowledge that any claims regarding the comprehensiveness of this parsimonious list are subject to empirical assessment. As an initial validation, I describe the un-problematic representation of integration patterns below and argue that for the purpose of this research—accessing design knowledge for enterprise integration—this parsimonious list is likely to be sufficient. Figure 5.3 provides representation of a few example integration patterns as a sequence of speech acts along with the rationale for the representation.
5.3.2.3 Characterizing tasks with action types

High-level business actions were obtained from different business activity behaviors described in the Unified Modeling Language (UML) specification (UML, 2005). UML is widely used for creating business process models and software application design solutions (Gardner, 2003). The UML specification describes about 54 different possible activities; each activity description was
interpreted with respect to their applicability to seven speech acts. From this review, I identified 16 different activities as relevant to this mapping. These 16 action types were transformed into their simplest forms, to make it easier to comprehend the action performed. For instance, the AcceptCallAction action type was found relevant for the Acknowledge speech act that was transformed into *Accept call with no receipt send*. After this transformation, I scoped 16 action types that were refined into 11 action types by aggregating the action types that could encompass other action types into single action types. For instance, the *Accept call with no receipt send* and *Accept event with no receipt send* action types that are relevant to the Acknowledge speech act were aggregated into the *Accept with no receipt send* action type. Table 5.2 provides list of 11 action types along with their associations to relevant speech acts.

Table 5.2  Mapping between speech acts and action types

<table>
<thead>
<tr>
<th>Action Types</th>
<th>Speech Acts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept with no receipt send</td>
<td>Acknowledge</td>
</tr>
<tr>
<td>Reject with no receipt send</td>
<td>Cancel</td>
</tr>
<tr>
<td>Invocation</td>
<td>Direct</td>
</tr>
<tr>
<td>Declare completion of task</td>
<td>Fulfill</td>
</tr>
<tr>
<td>Accept and send receipt</td>
<td>Inform</td>
</tr>
<tr>
<td>Provide information</td>
<td></td>
</tr>
<tr>
<td>Raise Exception</td>
<td></td>
</tr>
<tr>
<td>Reject and send receipt</td>
<td></td>
</tr>
<tr>
<td>Propose to perform task</td>
<td>Propose</td>
</tr>
<tr>
<td>Request for Information</td>
<td>Query</td>
</tr>
<tr>
<td>Request to cancel task</td>
<td></td>
</tr>
</tbody>
</table>

5.3.3  Inferring mapping between task and integration patterns with speech acts

An infrastructure that implements this mapping must require a knowledge base that captures the relationships and constraints among concepts such as tasks, action types, speech acts, and integration patterns (Sowa, 2000). Ontology is defined as “an explicit specification of a conceptualization” (Gruber, 1993), therefore, ontologies are useful in developing knowledge bases. Ontology allows the description of concepts and expresses relationships and constraints.
between concepts in a declarative formalism (Wang, Chan, & Hamilton, 2002). I developed ontological representations of my mechanism using Web Ontology Language (OWL) (OWL, 2004). OWL provides a standardized semantic mark-up language for publishing and sharing ontologies intended to be processed by applications (OWL, 2004). OWL supplies the required formal semantics to encode hierarchies of classes that describe concepts and relate these classes to each other using properties in a descriptive logic based constraint language (Kamthan & Pai, 2005; OWL, 2004). Figure 5.4 provides conceptual model for the OWL knowledge base.

![Figure 5.4. Conceptual model of the OWL knowledge base](image)

An infrastructure that implements this mapping would also require the capability to make inferences about appropriate integration patterns based on the business processes described using BPMN notations. The capability to make inferences using the OWL knowledge base is provided by OWL reasoners. I used the Bossam OWL Reasoner (Bossam, 2006) to make inferences about appropriate integration patterns for the given sets of tasks. Bossam is a RETE-based forward chaining rule engine with native supports for reasoning with OWL ontologies (Jang & Sohn, 2004). The Bossam inference engine translates OWL classes and restrictions as facts, thus, the relationships among action types, speech acts, and integration patterns are declared as sets of...
facts. A set of rules must be written to make inferences on the appropriate patterns based on these declared sets of facts.

In order to facilitate this inference, interactions in the business processes should be identified and declared as additional sets of facts to the Bossam OWL Reasoner. Running the Bossam OWL Reasoner with new fact sets provides mapping between interactions identified and sequences of speech acts that are appropriate for identified interactions. The rule that is used for inferring mapping between tasks in the interaction and speech acts is given below:

**Rule in the form required by Bossam**

If

\[
\text{bossam:allValuesFromDeclared}(?x, \text{hasSpeechAct}, ?s1) \text{ and hasActionType}(?t1, ?a1) \text{ and rdfs:subClassOf}(?a1, ?x) \text{ and not } \text{owl:disjointWith}(?a1, ?s1)
\]

Then

\[
\text{hasSpeechAct}(?t1, ?s1)
\]

**Rule in simplistic form**

If

\[
\text{Task } t1 \text{ has Action Type } a1 \text{ AND Action Type } a1 \text{ is associated with Speech Act } s1
\]

Then

\[
\text{Relevant Speech Act for Task } t1 \text{ is } s1
\]

The speech acts inferred for each task in the identified interactions are constructed as speech act sequences, which are then, declared as additional sets of facts to the Bossam OWL Reasoner. Running the Bossam OWL Reasoner with the current fact set provides mapping between the interactions identified and the integration patterns that are appropriate for the identified interactions. The partial rule set that is used for inferring mapping between the speech act sequences and integration patterns is given in the table 5.3.
Table 5.3   Partial set of enterprise integration pattern inference rules

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Inference Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregator</td>
<td><strong>Rule in the form required by Bossam</strong></td>
</tr>
<tr>
<td></td>
<td>If</td>
</tr>
<tr>
<td></td>
<td>hasAssociation(?ind, Many_to_One) and hasSenderSA(?x, ?y, ?z) and</td>
</tr>
<tr>
<td></td>
<td>hasReceiverSA(?a, ?b, ?c) and hasSender(?ind, ?t1, ?y) and hasReceiver(?ind,</td>
</tr>
<tr>
<td></td>
<td>?t2, ?b) and numConn(?y, ?b, ?n) and [?n = 1] and [?z = Inform] and [?c =</td>
</tr>
<tr>
<td></td>
<td>Inform] then</td>
</tr>
<tr>
<td></td>
<td>hasPattern(?ind, Aggregator)</td>
</tr>
<tr>
<td></td>
<td><strong>Rule in simplistic form</strong></td>
</tr>
<tr>
<td></td>
<td>If</td>
</tr>
<tr>
<td></td>
<td>Interaction ind has Many to One association between initiator y and</td>
</tr>
<tr>
<td></td>
<td>responder b AND Relevant speech act for action performed by Initiator(s) y is</td>
</tr>
<tr>
<td></td>
<td>Inform AND Relevant speech act for action performed by Responder b is Inform</td>
</tr>
<tr>
<td></td>
<td>Then</td>
</tr>
<tr>
<td></td>
<td>Appropriate pattern for identified Interaction ind is Aggregator</td>
</tr>
<tr>
<td>Command Message</td>
<td><strong>Rule in the form required by Bossam</strong></td>
</tr>
<tr>
<td></td>
<td>If</td>
</tr>
<tr>
<td></td>
<td>hasAssociation(?ind, One_to_One) and hasSenderSA(?x, ?y, ?z) and</td>
</tr>
<tr>
<td></td>
<td>[?z = Direct] and hasReceiverSA(?a, ?b, ?c) and hasSender(?ind, ?t1, ?y) and</td>
</tr>
<tr>
<td></td>
<td>hasReceiver(?ind, ?t2, ?b) and numConn(?y, ?b, ?n) and [?n = 1]</td>
</tr>
<tr>
<td></td>
<td>then</td>
</tr>
<tr>
<td></td>
<td>hasPattern(?ind, Command_Message)</td>
</tr>
<tr>
<td></td>
<td><strong>Rule in simplistic form</strong></td>
</tr>
<tr>
<td></td>
<td>If</td>
</tr>
<tr>
<td></td>
<td>Interaction ind has One to One association between initiator y and responder b</td>
</tr>
<tr>
<td></td>
<td>AND Relevant speech act for action performed by Initiator(s) y is Direct</td>
</tr>
<tr>
<td></td>
<td>Then</td>
</tr>
<tr>
<td></td>
<td>Appropriate pattern for identified Interaction ind is Command Message</td>
</tr>
<tr>
<td>Point-to-Point Channel</td>
<td><strong>Rule in the form required by Bossam</strong></td>
</tr>
<tr>
<td></td>
<td>If</td>
</tr>
<tr>
<td></td>
<td>hasAssociation(?ind, One_to_One) and hasSenderSA(?x, ?y, ?z) and</td>
</tr>
<tr>
<td></td>
<td>hasReceiverSA(?a, ?b, ?c) and hasSender(?ind, ?x, ?y) and hasReceiver(?ind,</td>
</tr>
<tr>
<td></td>
<td>?a, ?b) and numConn(?y, ?b, ?n) and [?n = 1]</td>
</tr>
<tr>
<td></td>
<td>then</td>
</tr>
<tr>
<td></td>
<td>hasPattern(?ind, Point-to-Point Channel)</td>
</tr>
<tr>
<td></td>
<td><strong>Rule in simplistic form</strong></td>
</tr>
<tr>
<td></td>
<td>If</td>
</tr>
<tr>
<td></td>
<td>Interaction ind has One to One association between initiator y and responder b</td>
</tr>
<tr>
<td></td>
<td>Then</td>
</tr>
<tr>
<td></td>
<td>Appropriate pattern for identified Interaction ind is Point-to-Point Channel</td>
</tr>
</tbody>
</table>

5.3.4 Surfacing conflicts among integration patterns

The mapping described above allows designers to decompose business processes into smaller units, or interactions, and then identify appropriate integration patterns for each of these smaller
In order to develop complete integration solutions, designers would need to assemble identified integration patterns according to the interaction sequence in the business processes. However, such a design by assembly requires an intelligent aggregation of patterns that resolves constrains and conflicts among the patterns (Purao et al., 2003).

Each pattern has its own nature, proper place of use, and associated constrains (Vokáč, Tichy, Sjöberg, Arisholm, & Aldrin, 2004). Conflicts among selected patterns occur when the constrains of one pattern affect the functioning of another pattern (Baudry, Traon, Sunye, & Jezequel, 2001). Such conflicts lead to internally inconsistent integration solutions (Alencar, Cowan, Dong, & Lucena, 1999). Therefore, conflicts among selected patterns must be identified and resolved. Conflicts among selected patterns can also occur due to the availability of patterns that can replace that set of selected patterns (Mens & Tourwe, 2001). Surfacing conflicts cause by the availability of patterns with superior structure is important, because systems with such conflicts will drift away from their intended structure after several iterations (Mens & Tourwe, 2001). Resolving conflicts among selected patterns will be conducted as a part of future work.

5.3.5 Demonstration through an example

To demonstrate the application of the mapping developed in this research, I used a supply chain management scenario (SCM Scenario, 2003) developed by the Web Services Interoperability Organization (WS-I). The specific business process I identified for this demonstration is customer order processing. In this process, a Customer places an order at a retailer’s website, the Retailer requests Warehouses to send stock details for the Customer’s order items. The Retailer
then prepares and returns a quote for the Customer’s order. Figure 5.5 shows the business process diagram for customer order processing drawn with BPMN notations.

![Business process diagram for customer order processing](image)

**Figure 5.5. Business process diagram for customer order processing**

The following interactions can be identified in this process: (1) one-to-one interaction, with tasks 1 and 3; (2) One-to-many interaction, with tasks 3, 4, 5, and 6; (3) Many-to-one interaction, with tasks 4, 5, 6 and 7; and (4) One-to-one interaction, with tasks 9 and 11. The appropriate speech act is identified for each task in the identified interactions based on their action type. These identified interactions, tasks, and their action types are declared as facts to the Bossam OWL Reasoner. Running Bossam OWL Reasoner with the current fact set provided mapping between tasks and relevant speech acts, the results of this mapping are provided in table 5.4 below. For instance, the action type for task 1 is *Request for information* because the customer is sending a customer order requesting a quote. The associated speech act for the Request for information action type is Query, hence, the Bossam OWL Reasoner mapped task 1 to the Query speech act.
Table 5.4  Speech act sequence for identified interactions

<table>
<thead>
<tr>
<th>Interactions</th>
<th>Tasks</th>
<th>Performer</th>
<th>Role</th>
<th>Action Type</th>
<th>Speech act</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Customer</td>
<td>Initiator</td>
<td>Request for information</td>
<td>Query</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Retailer</td>
<td>Responder</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Retailer</td>
<td>Initiator</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Warehouse A</td>
<td>Responder</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Warehouse B</td>
<td>Responder</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Warehouse C</td>
<td>Responder</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>Warehouse A</td>
<td>Initiator</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Warehouse B</td>
<td>Initiator</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Warehouse C</td>
<td>Initiator</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Retailer</td>
<td>Responder</td>
<td>Accept and no receipt</td>
<td>Acknowledge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>Retailer</td>
<td>Initiator</td>
<td>Provide information</td>
<td>Inform</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Customer</td>
<td>Responder</td>
<td>Accept and no receipt</td>
<td>Acknowledge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The speech acts of the initiator and responder form a sequence of speech acts for each interaction. These speech act sequences are declared as facts to the Bossam OWL Reasoner. Running the Bossam OWL Reasoner with the current fact set provided mapping between the interactions identified and the integration patterns that are appropriate for the identified interactions. Below, table 5.5 presents the results of this mapping. For example, interaction 3 is a many-to-one interaction (4, 5, 6, and 7), with the Inform speech act for initiators and the Acknowledge speech act for responders, therefore, the Bossam OWL Reasoner suggested the Aggregator pattern as appropriate for interaction 3.

Table 5.5  Integration patterns for identified interactions

<table>
<thead>
<tr>
<th>Interactions</th>
<th>Speech act sequence</th>
<th>Patterns Identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Query, Inform</td>
<td>Point-to-Point Channel Document Message</td>
</tr>
<tr>
<td>2</td>
<td>Inform, Inform, Inform, Inform, Inform</td>
<td>Recipient List</td>
</tr>
<tr>
<td>3</td>
<td>Inform, Inform, Inform, Acknowledge</td>
<td>Aggregator</td>
</tr>
<tr>
<td>4</td>
<td>Inform, Acknowledge</td>
<td>Point-to-Point Channel Document Message</td>
</tr>
</tbody>
</table>

In order to develop consistent solutions, conflicts among these selected integration patterns must be surfaced. For instance, in this example, interactions 1 (tasks 1 and 3) and 4 (tasks 9 and 11) have two-way communication with role reversal, in other words, for interaction 1, the Customer
is the initiator and the Retailer is the responder, while for interaction 4, the Customer is the responder and the Retailer is the initiator. These two interactions follow the request-reply format, thus, the Request-Reply Pattern should be selected for these interactions instead of two different Point-to-Point Channel patterns.

5.4 Assembling integration patterns into web service solutions

In order to develop complete integration solutions, designers need to assemble selected integration patterns to construct implementable integration solutions (Alencar et al., 1999; Purao et al., 2003). However, integration patterns are abstract and non-implementable solutions. Therefore, these patterns need to be converted into appropriate web service specifications. Web service conversation specifications are an appropriate type of specification for implementing integration patterns (Umapathy & Purao, 2006).

Currently, there are three web service conversation specifications: Web Services Choreography Description Language (WS-CDL) (WS-CDL, 2005), Conversation Policy for XML (cpXML) (cpXML, 2002), and Web Services Conversation Language (WSCL) (WSCL, 2002). A comparative evaluation of these specifications against the requirements for supporting long-running business conversations suggests that cpXML is more effective than the other two (Umapathy, 2006). Therefore, the objective of this research is to translate the integration solutions represented in the form integration patterns into cpXML specifications.

A prerequisite for conversation specifications is a “conversation model” of the integrated systems. A conversation model describes the reactive behavior of web services based on a
“conversation context” (Benatallah, Casati et al., 2003; Hanson, Nandi, & Kumaran, 2002), and is specified as a state-transition diagram. For each conversation state, the model describes sets of services that are part of the interaction and messages that need to be exchanged among them. Thus, the conversation model describes multi-step exchanges of correlated messages that need to be performed by participating services in order to achieve a business goal (Ardissono et al., 2003). Therefore, a prerequisite for generating web service conversation specifications is development of conversation model of an integrated system.

One possible strategy is to convert the assemblage of individual integration patterns that are used to construct integration solutions (Umapathy & Purao, 2006). This strategy is plausible because business process models and conversation models are syntactically similar, given that business process models provide the order of interactions among participating applications in the form of sequences of tasks executed by participating applications (Q. Chen & Hsu, 2002). The individual integration patterns that were selected as design solutions are assembled according to the task sequence prescribed in the business processes to construct integration solutions. Moreover, each integration pattern used in the integration solution represents message exchanges performed by integrated applications (Hohpe & Woolf, 2004). Therefore, each integration patterns is considered a conversation state and integration patterns that follow in the sequence are considered transition states. Message exchanges represented by each integration pattern are considered messages to be exchanged between conversation states. The assemblage of individual integration patterns in accordance to business processes, consequently form conversation models.
5.4.1 Generating conversation specifications

In order to facilitate the generation of conversation specifications, the composition of integration patterns, or conversation models, should be developed using an appropriate representation model. I developed a simple XML schema to represent the composition of enterprise integration patterns (ceipXML) for the purpose of describing, storing, and retrieving conversation models. This schema, ceipXML, includes specifications such as sender, receiver, message channel, message schema details, and sequencing; and is free from business logic and conversation state details. Figure 5.6 shows a simplified version of the ceipXML specification. The schema of the ceipXML specification can be accessed via Internet (eipXML Schema, 2006). Because both ceipXML and cpXML (conversation specification) are XML documents, cpXML can be generated through XML transformation on ceipXML using the appropriate style sheet language (XSLT, 1999). Figure 5.7 depicts mapping between ceipXML and cpXML specifications. Figure 5.8 shows mapping between partial ceipXML and cpXML specifications for the SCM scenario example described earlier in the section 5.3.5.

```xml
<ComposedEIP>
  <Name/>
  <PatternOrder>
    <Order>
      <Pattern/>
    </Order>
  </PatternOrder>
  <Pattern Name = "" PatternType = "">
    <Sender/>
    <Receiver/>
    <Message>
      <Encoding/>
      <Schema/>
    </Message>
  </Pattern>
</ComposedEIP>
```

Figure 5.6. Simplified skeleton of ceipXML specification
Figure 5.7. Mapping between ceipXML and cpXML specifications

Figure 5.8. Partial ceipXML and cpXML mapping for SCM scenario
5.5 Research prototype

Integration designers are required to perform complex and diverse tasks such as gathering requirements, designing, analyzing, and implementing integration solutions (Frank, 2002). As these are highly repetitive and complex tasks, integration designers require step-by-step guidance for performing these design tasks (Frank, 2002). The research outcomes described above have been compiled into sets of steps that can act as guidance for developing appropriate integration solutions using enterprise integration patterns and web services. The guidance steps are divided into three phases encompassing six steps as shown in figure 5.9. Based on the above research outcomes and these guidance steps, I have developed a research prototype called IDAssist (Integration Designer Assistant).

![Diagram of Process for developing integration solutions]

**Figure 5.9.  Process for developing integration solutions**

Figure 5.10 below shows the architecture of the research prototype. It consists of the following components: business process diagram editor, interaction extraction, pattern inference, pattern knowledge base, process file store, EIP generator, EIP store, CP generator, and CP store.
The business process diagram editor component allows designers to develop business processes that provide integration requirements. This capability is provided in the form of a graphical interface (see figure 5.11), which allows designers to use BPMN notations (BPMN, 2006) to develop business process diagrams using a drag and drop function (see step 1 in the figure 5.9). The business processes developed by designers are then converted into an XPDL specification (XPDL, 2005) and stored in the process files store.

Figure 5.10. Architecture of the Integration Designer Assistant

Figure 5.11. Business process diagram editor interface
The interactions extraction component traverses through the business processes to identify and list all possible interactions among actors (see step 2 in the figure 5.9). For each identified interaction, the pattern inference component identifies integration pattern(s) (see step 3 in the figure 5.9). This component uses the pattern knowledge base developed using OWL, which contains heuristics to select appropriate patterns. This component uses Bossam OWL Reasoner (Bossam, 2006) to make inferences about appropriate integration patterns based on the business process elements and heuristics in the knowledge base. The Bossam inference engine translates OWL classes and restrictions as facts. These facts are used to make inferences about appropriate patterns along with sets of rules.

The EIP generator component composes the selected integration patterns (as specified in the XPDL files in the process files store) into a conversation model (see step 5 in the figure 5.9). This component generates a ceipXML document that describes the composed enterprise integration patterns as an integrated system solution and stores it in the EIP store. Finally, the CP generator component uses the ceipXML document to generate cpXML specifications (see step 6 in the figure 5.9), and stores it in the CP store. Figure 5.12 provides mapping between software and research components.
5.6 Validation against design science research guidelines

One of the main objectives of this research is to develop a conceptual model for accessing integration solutions and to translate those solutions into web service specifications. This conceptual model is implemented through the construction of a software artifact that assists integration designers with the development of integration solutions. Thus, the software artifact provides heuristic information for designers to develop integration solutions. This heuristics information should be relevant and applicable (Brocke & Buddendick, 2006) and also should be
specified and constructed using the appropriate theoretical foundations (Weber, 2003). The design science research paradigm (Hevner et al., 2004) provides guidelines to strengthen the construction of information technology artifacts that are theoretically and empirically grounded (Broke & Buddendick, 2006). Therefore, this research uses design science research guidelines as its research methodology. Table 5.6 provides seven guiding principles for conducting design science research (Hevner et al., 2004). These principles describe what constitutes good (i.e., rigorous and relevant) design science research, therefore, design science researchers should address these principles in some manner (Carlsson, 2006; Hevner et al., 2004). Figure 5.13 depicts a summary of this research effort in accordance with the seven guiding principles.

<table>
<thead>
<tr>
<th>Guideline</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guideline 1: Design as an Artifact</td>
<td>Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.</td>
</tr>
<tr>
<td>Guideline 2: Problem Relevance</td>
<td>The objective of design-science research is to develop technology-based solutions to important and relevant business problems.</td>
</tr>
<tr>
<td>Guideline 3: Design Evaluation</td>
<td>The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.</td>
</tr>
<tr>
<td>Guideline 4: Research Contributions</td>
<td>Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.</td>
</tr>
<tr>
<td>Guideline 5: Research Rigor</td>
<td>Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.</td>
</tr>
<tr>
<td>Guideline 6: Design as a Search Process</td>
<td>The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.</td>
</tr>
<tr>
<td>Guideline 7: Communication of Research</td>
<td>Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.</td>
</tr>
</tbody>
</table>
Figure 5.13. Summary of research effort in accordance to design science guidelines
CHAPTER 6  EMPIRICAL EVALUATION

Research hypothesis: LAP is an appropriate theoretical framework for web services

Evaluate Appropriateness

RQ1
First Essay
Focus: Architecture Level

RQ2
Second Essay
Focus: Conceptual Level

RQ3
Third Essay
Focus: Empirical Level
6.1 Need for empirical evaluation

Design science principles suggest that the theoretical constructs embodied in Information Technology (IT) artifacts should be evaluated to demonstrate the appropriateness and utility of their usage (Hevner et al., 2004). Therefore, the appropriateness and utility of LAP constructs used in the conceptual model (i.e., the mapping between integration requirements and integration patterns developed in the chapter 5) should be assessed. The appropriateness and utility of LAP constructs can be demonstrated by gauging the effectiveness of solutions provided through empirical methods (Dumay et al., 2005).

The appropriateness and utility of LAP constructs can be demonstrated by establishing a cause and effect relationship between LAP constructs and the effectiveness of integration solutions (Tvedt & Collofello, 1995). Cause and effect relationships can be explained by showing that solutions developed with the aid of LAP constructs are better than solutions developed without the aid of LAP constructs (Eickelmann & Richardson, 1996). Controlled experimentation (Shadish et al., 2001) is the preferred strategy to determine these cause and effect relationships (Sjoeberg et al., 2005).

The research prototype developed in chapter 5 produces web service conversation specifications as solutions for integrating systems. Conversation specifications are developed based on integration solutions, thus, the development of conversation specifications is affected by integration solutions. Additionally the development of integration solutions is affected by the selection of appropriate integration patterns, because the selection of incorrect patterns or patterns that constrain the interactions of other patterns leads to internally inconsistent designs.
(Alencar et al., 1999). Therefore, accessing and identifying appropriate integration patterns directly affects the development of conversation specifications. For those reasons, the objective of this evaluation is to determine whether integration solutions developed with the aid of LAP constructs are better than those solutions developed without the aid of LAP constructs.

6.2 Research hypothesis

Design knowledge represented in the form of patterns, such as integration patterns, is claimed to be effective for capturing, communicating, and reusing the best design solutions (Beck et al., 1996). A pattern captures the rationale and trade-offs behind the usage of proven design solutions (Keller, Schauer, Robitaille, & Pagé, 1999). Such patterns, typically include descriptions of (i) the problem context, (ii) the possible solution, (iii) the forces that enable or constrain possible solutions to the problem, (iv) the resulting context after usage of the solution, and (v) examples of the solution (Carroll & Farooq, 2007).

In the software engineering domain it is claimed that a well-documented pattern, along with its rationale and trade-offs, enhances the quality of design solutions because such patterns address the fundamental challenges of software design (Schmidt, 1995). These challenges include the communication of proven design knowledge, enabling the reuse of solutions, improving communication among designers and developers, helping avoid errors committed by other designers, and facilitating the transfer of new design knowledge (Schmidt, 1995). The use of patterns is believed to subdue the above challenges, thus, aiding the production of better design solutions (Beck et al., 1996; Prechelt, Unger-Lamprecht, Philippsen, & Tichy, 2002).
In the context of enterprise integration, designing an integrated system is a process of analyzing integration requirements to produce design solutions that will serve as a basis for implementing that system. Designing integration solutions, therefore, is a process of finding appropriate integration patterns for each identified integration requirement. Finding appropriate patterns for a given problem is a matter of discovery, or accessing and selecting an appropriate pattern (Schmidt, Fayad, & Johnson, 1996). Hence, the selection of appropriate integration patterns should produce better integration solutions.

The purpose of this evaluation is to demonstrate the appropriateness and utility of the LAP constructs embedded in the design support for accessing and selecting the appropriate integration patterns for designing integration solutions. Therefore, the utility of LAP constructs is shown by comparing the effectiveness of design support with the aid of LAP constructs against design support without the aid of LAP constructs in producing better integration solutions.

6.2.1 Design support for accessing integration patterns

Each integration pattern has its own terms of use along with constraints and tradeoffs (Vokáč et al., 2004; Wasserman, 1996), therefore, the improper use of a pattern leads to design defects that can potentially cause failure during the execution of an integrated system (Alencar et al., 1999; Prechelt, 2001). These design defects can be measured by detecting design errors in the solutions (Wasserman, 1996). The effectiveness of the design support for accessing and identifying patterns, consequently, can be measured in terms of the amount of design errors produced (Jankowski, 1997). For that reason, an effective design support has a higher likelihood of avoiding design errors while producing better design alternatives (Girczyc & Carlson, 1993).
order to demonstrate the appropriateness and utility of LAP constructs used in the research prototype, it is necessary to show that design support with LAP constructs produces fewer design errors than design support without LAP constructs. This proposition is reflected in the first hypothesis (H1) provided below.

H1: Designs produced with the support of LAP constructs contain fewer design errors than designs produced without the support of LAP constructs.

Typically, when designing integration solutions, designers need to decompose the integration problem into smaller units, then identify appropriate patterns for each of these smaller units, and re-assemble patterns to construct solutions meeting the integration requirements (Alencar et al., 1999; Purao et al., 2003). Such a design by assembly requires the intelligent combining of patterns in order to resolve the constraints and conflicts among the patterns (Purao et al., 2003). This task reflects the considerable design effort required for integrating disparate systems (Dellarocas, 1997). Thus, in order to demonstrate the appropriateness and utility of LAP constructs used in the research prototype, it is necessary to show that design support with LAP constructs require less design effort for developing solutions when compared to design support without LAP constructs. This proposition is reflected in the second hypothesis (H2) provided below.

H2: Designs produced with the support of LAP constructs require less design effort than designs produced without the support of LAP constructs.
6.2.2 Dealing with complexity

Design errors and design effort can also be influenced by the complexity of the integration problem (Basili & Perricone, 1984). The complexity of the integration problem is primarily concerned with identifying the appropriate design knowledge to integrate dissimilar system functionalities (Li & Williams, 2002). As the complexity of the integration problem increases, the required design effort also increases (Dellarocas, 1997) along with the possibility of design errors (Jankowski, 1997). Therefore, complexity of the integration problem is considered a control variable in the experimental evaluation model.

6.2.2.1 Task complexity

The complexity of the integration problem influences the complexity associated with the task of searching for and applying the appropriate design knowledge to address identified integration requirements. While there is no consensus on the definition of task complexity, it is generally considered to be the degree of cognitive load or mental effort required to identify and/or solve a problem (Jung, Olfman, Ryan, & Park, 2005). According to Campbell (Campbell, 1988), information overload, information diversity, and the rate of information changes are factors that contribute to task complexity. The fundamental characteristics that contributes to increase in task complexity are (Campbell, 1988; P. Xu, 2005): (i) a large number of possible paths to achieve a desired outcome, (ii) a large number of possible outcomes for the task, (iii) conflicts among desired outcomes, (iv) the uncertainty associated with required activities and desired outcomes and (v) ambiguous and ill-structured tasks.
Wood (Wood, 1986) reviewed various theoretical studies on tasks and suggested that task complexity is a function of the number of actions that must be performed and the number of information cues that must be processed when performing a task (Jung et al., 2005). Therefore, tasks that require performing a large number of actions while processing a large number of information cues are considered complex tasks and those tasks that require a lesser number of actions and information cues are simple tasks.

In the context of enterprise integration, the complexity of an integration problem is expressed in terms of the number of integration points, or interactions in the business processes (Lam & Shankararaman, 2004). Therefore, a complex integration problem with many integration points has characteristics of multiple paths to desired outcomes, multiple desired outcomes, and conflicts and ambiguity among the desired outcomes. A complex integration problem, thus, requires more actions to be performed while processing more information cues than does a simple integration problem. This is why I argue that the need for design support with LAP constructs should be more pronounced for complex integration problems than for simple integration problems. However, it is possible that as the designer engages with the sequential decision making process, some learning will occur (Joshi, 2001). Such learning could possibly reduce effort required to process information cues, thus, provide difficulties in measuring difference between complex and simple problems.

Based on the discussion above, the following hypotheses are proposed. Designed solutions for a complex integration problem are more error prone than those for a simple integration problem (Jankowski, 1997), therefore, it is necessary to show that design support with LAP constructs
maintains its ability to reduce design errors when the complexity of the integration problem is increased. This proposition is reflected in the third hypothesis (H3) provided below.

**H3:** The use of design support with LAP constructs leads to greater reduction in design errors for complex integration problems than for simple integration problems.

Design effort, with respect to integrating disparate systems, lies in identifying the appropriate integration tactics for integrating systems as well as managing the interdependencies and mismatches among identified tactics (Dellarocas, 1997). As the complexity of the integration problem increases, the required design effort for developing quality solutions also increases considerably (Lam & Shankararaman, 2004). Therefore, it is necessary to show that design support with LAP constructs significantly alleviates the level of design effort needed for developing quality solutions as the complexity of the problem increases. This proposition is reflected in the fourth hypothesis (H4) provided below.

**H4:** The use of design support with LAP constructs leads to greater reduction in design effort for complex integration problems than for simple integration problems.

Design error and design effort can also be affected by the designer’s characteristics (Wrigley & Dexter, 1991). Interactions among the designer’s characteristics, design error, and design effort, therefore, could affect the evaluation. Therefore, I acknowledge the designer’s characteristics as a factor that needs to be considered during the analysis of this data. Table 6.1 provides a summary of the factors under study in this evaluation.
Table 6.1  Summary of factors under study

<table>
<thead>
<tr>
<th>Design support</th>
<th>Integration Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design support with LAP constructs (X)</td>
<td>Simple X1</td>
</tr>
<tr>
<td>Design support without LAP constructs (Y)</td>
<td>Complex X2</td>
</tr>
<tr>
<td></td>
<td>Y1</td>
</tr>
<tr>
<td></td>
<td>Y2</td>
</tr>
</tbody>
</table>

This empirical evaluation tests for four hypotheses, as depicted in the figure 6.1. The first hypothesis tests the utility of the design support with LAP constructs in developing quality solutions, by testing whether designs produced by X contain fewer design errors than designs produced by Y. The second hypothesis tests the claim that the design support with LAP constructs also reduces the design effort required of designers to produce quality solutions, by testing whether designs produced by X require less design effort than designs produced by Y. The third hypothesis tests the claim that the design support with LAP constructs is more useful for developing solutions with less design errors for complex problems than simple problems, by comparing the difference in the amount of design errors produced between Y1-X1 and Y2-X2. The fourth hypothesis tests the claim that the design support with LAP constructs is more useful in developing solutions with less design effort for complex problems than for simple problems, by comparing the difference in the amount of required design effort between Y1-X1 and Y2-X2.

![Figure 6.1. Conceptual model of hypotheses to be tested](image-url)
6.3 Experimental Design

In order to test the first and second hypothesis, or in other words to perform comparative tests on design support with LAP constructs against design support without LAP constructs, the research prototype should facilitate the development of integration solutions with the aid of LAP constructs and also without the aid of LAP constructs. Therefore, two versions of the research prototype were developed, one with ability to design integration solution with the support of LAP constructs and other without support of LAP constructs. In order to test the third and fourth hypotheses, integration solutions should be developed in the context of either complex or simple enterprise integration problems.

For that reason these research hypotheses are tested in either a complex or simple integration problem context using the research prototype to design integration solutions either with the aid of LAP constructs or without the aid of LAP constructs. These research hypotheses are, therefore, tested through an experiment with two design support levels (the presence or absence of the LAP constructs) and two integration problem levels (either simple or complex). This structure gives the experiment an appropriate 2x2 factorial design (Barton, 1999), as illustrated in the table 6.2.

<table>
<thead>
<tr>
<th>Design support with LAP constructs</th>
<th>Design support without LAP constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Complex</td>
</tr>
<tr>
<td>Group 11</td>
<td>Group 12</td>
</tr>
<tr>
<td>Group 21</td>
<td>Group 22</td>
</tr>
</tbody>
</table>

6.3.1 Independent variables

The independent (control) variables of the experiment are the design support type and the complexity level of the integration problem. Design support has two levels: design support with LAP constructs and design support without LAP constructs. Design support levels are controlled
by developing integration solutions using two different research prototype versions—one with the support of LAP constructs and other without the support of LAP constructs.

There are two levels of integration problem complexity: simple integration problems and complex integration problems. Designing integration solutions is essentially about connecting integration points using appropriate integration patterns, therefore, enterprise integration problems are $N^2$ complexity problems (Hohpe & Woolf, 2004). Thus, $N$ integration points lead to $N(N-1)/2$ possible connections. For instance, an integration problem with ten integration points could possibly require 45 or less connections to be made. However, the review of prior research did not provide a distinct number of integration points to distinguish between simple and complex integration problems. In order to facilitate this evaluation, a pilot study (see section 6.6) was conducted to identify the appropriate number of integration points to distinguish between simple and complex problems.

### 6.3.2 Dependent variables

The purpose of the experiment is to evaluate the effectiveness of the design support embedded in the research prototype and this effectiveness is measured in terms of design error and design effort. Therefore, the dependent variables for the experiment are design error and design effort. Below, the methods for measuring these two dependent variables are discussed.

#### 6.3.2.1 Design error

Measuring design errors is a complex task because there are no automated means for detecting design defects (Tahvildar & Kontogiannis, 2004). One possible way to measure design errors is,
therefore, to evaluate solutions developed through research prototypes against those that an
expert designer might develop (Purao et al., 2003). To use this approach, an expert designer
needs to develop integration solutions using integration patterns for the problems that are to be
used in the experiments. These solutions developed by an expert designer will be used to perform
comparative evaluations on the solutions developed through the experiment to identify design
errors.

The comparative evaluation can be performed based on well established means for measuring
errors in the presence or absence of the experimental model, such as false negatives (FN) and
false positives (FP) (Fielding & Bell, 1997). False negatives are known as Type I errors that
measure the completeness of solutions (Moody, Sindre, Brasethvik, & Sølvberg, 2002). Type I
errors are omission errors such as requirements with no or missing patterns (Purao et al., 2003).
False positives are known as Type II errors that measure of the correctness of the solutions
(Moody et al., 2002). Type II errors are commission errors such as requirements with incorrect
patterns (Purao et al., 2003). Additionally, there are inconsistency errors that are known as Type
III errors, such as requirements with inferior patterns (i.e., patterns with similar structures that
can improve performance were ignored) (Moody et al., 2002). Errors of Type II and Type III are
classified as major and minor. Errors that may seriously affect the execution of the integration
solutions are considered major errors, thus, any other errors are considered minor. Major errors
are penalized three points while minor errors are penalized one point. All Type I errors are
classified as major errors, because requirements with missing patterns severely undermines the
execution of the integration solutions. Table 6.3 provides a summary of the scheme for assessing
design errors along with examples of minor and major errors for each error type.
### Table 6.3  Design error assessment scheme

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Error Categorization</th>
<th>Minor</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type I (omission errors)</td>
<td></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Example</td>
<td>All errors of Type I are major errors</td>
<td>Pattern missing for identified integration requirement</td>
<td></td>
</tr>
<tr>
<td>Type II (commission errors)</td>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Example</td>
<td>Using Recipient List pattern instead of Publish-Subscribe pattern</td>
<td>Using Splitter pattern instead of Aggregator pattern</td>
<td></td>
</tr>
<tr>
<td>Type III (inconsistency errors)</td>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Example</td>
<td>Using two independent Point-to-Point channel patterns instead of Request-Reply pattern</td>
<td>Using independent Point-to-Point channel patterns for each data types instead of Datatype channel pattern</td>
<td></td>
</tr>
</tbody>
</table>

#### 6.3.2.2 Design effort

The estimation of required design effort for developing information systems is not a widely researched topic (Roy, Kelvesjo, Forsberg, & Rush, 2001). Design effort is heavily influence by the designer’s experience and knowledge, design support provided, and the complexity of the problem (Wrigley & Dexter, 1991). Design effort is considered to be the competence level required for applying a method (Moody, 2003). Typically, the design effort required for a given problem is measured in terms of the time taken and the number of steps required to develop solutions (Bocco, Moody, & Piattini, 2005). The research prototype records each step (i.e., task) performed by the subjects along with the time taken for task completion. Examples of tasks are viewing an identified interaction, viewing a pattern, selecting a pattern, and unselecting a pattern.

#### 6.4  Subject selection

The subjects for this experiment need to have a prerequisite knowledge of the enterprise integration domain. Therefore, subjects were recruited from a pool of students enrolled in an
enterprise integration course in a large University in the United States. Students enrolled in this course are trained to design integration solutions using a variety of integration techniques.

Students were selected as subjects due to their convenient accessibility for the researcher. Recruiting enterprise integration course students as subjects helps to control the influence of designer characteristics on the dependent variables. However, the selection of students as experimental subjects has been criticized because it introduces bias and the setting is unrealistic (Tichy, 2000). The differences between conducting experiments with students as opposed to professionals are: (i) the subjects’ experience and skills, (ii) the use of professional methods and tools, and (iii) team work versus individual work (Sjøberg, Arisholm, & Jørgensen, 2001). The first difference can be minimized by providing training to students on designing integration solutions using integration patterns (Carver, Jaccheri, Morasca, & Shull, 2003; Fredericks, 2007). The second and third differences are not applicable to this experiment as it uses research prototype developed by the researcher and does not involve team work. Several studies have explored the issue of using students (who are trained in the domain of the experiment) as subjects against professionals and have not found any substantial difference in the results (Höst, Regnell, & Wohlin, 2000; Sjøberg et al., 2005).

Moreover, students as subjects are considered appropriate for exploratory or illustrative experimental situations (Ferber, 1977; Sjøberg et al., 2005; Tichy, 2000). Several studies that are similar to this experiment such as Purao 2002 (Purao, Rossi, & Bush, 2002) and Sen 1997 (Sen, 1997) have used students for designers. Students are fitting subjects for this experiment, because the objective of this study is to demonstrate the appropriateness and utility of LAP constructs to
solve web service problems. Thus, the use of students as subjects does not have any external validity threat to my research (Land, Tan, & Bin, 2005).

### 6.5 Experimental procedures

Each subject was requested to solve two different problems (simple and complex) with two different versions of the research prototype (with LAP constructs and without LAP constructs). Each subject was first requested to solve a problem using the research prototype—for instance solving a simple problem with the LAP constructs version. After completion of the first problem, they were asked to solve another problem using another version of the research prototype—for instance solving a complex problem without the LAP constructs version. A detailed business process model was developed for each problem and provided to the subjects. Each subject was requested to design integration solutions for the given integration problem by accessing and selecting appropriate integration patterns with the aid of the research prototype.

Each subject was provided with detailed instructions (along with the corresponding screen shots of the research prototype) for designing integration solutions for the given problem using the given research prototype. A sample of the detailed instructions that were given to the subjects is provided in Appendix B. Post questionnaires (see Appendix C) were employed to gather information on the subjects’ skill sets, demographics, software design experience, confidence in the solutions they developed, and their thoughts of the research prototype. Below is a concise list of tasks that were performed by subjects in order to design integration solutions for a given problem using the research prototype:

1. Download and run the research prototype from the given link.
2. Select the given business process problem in the process selection window.
3. Browse and understand each integration requirement (i.e., interactions) listed.
4. Run the pattern reasoner.
5. Browse and understand the integration patterns listed for each interaction.
6. Select the appropriate integration patterns for each identified interaction.
7. View and submit the design summary report that includes the interactions list and the selected list of integration patterns for each of the identified interactions.

6.6 Pilot study

A pilot study was conducted to test and improve the experimental protocols used in this study.

The objectives of the pilot study were to:

- Identify runtime problems, if any present
- Identify the appropriate number of interactions in a business process to distinguish between simple and complex integration problems based on timelines for developing solutions and completing the study
- Ensure the experimental procedure and materials (e.g., the integration problem sets, the instructional documents, the experimental timeline, and the usage of IDAssist) are appropriate for the subsequent final experiments.

The pilot study was conducted using four business process problems with 5, 10, 15 and 21 interactions in order to identify the appropriate number of interactions to distinguish between large and small integration problems. Eight participants took part in the pilot study; each of them solved two different problems using two different research prototype versions. The pilot study revealed that for a problem with 21 interactions it takes more than one hour to develop complete solutions (i.e., consider all listed patterns and select the appropriate patterns for each interaction). Because I wished to keep the experimental timeline within one hour, for the final experiment a problem with 5 interactions is considered a simple problem while a problem with 15 or more interactions as a complex problem. The pilot study also revealed that the research prototypes are free of defects and the experimental protocols are easy to understand to perform the requested tasks.
6.7 Final experiment

The research sample for this experiment consists of 69 participants who were students enrolled in an “Advanced Enterprise Integration” course in a large University during the Fall 2007 semester. The advanced enterprise integration course provides a deep understanding of integration techniques across multiple application settings. A prerequisite for this class is a “Fundamentals of Systems and Enterprise Integration” course that imparts the required skills for planning, developing, implementing, and managing the integration of systems.

The advanced enterprise integration course had three sections, the first two sections of the course were taught by one instructor, while the third was taught by a different instructor. The experiment was conducted in all three sections and students were encouraged to voluntarily participate by offering them extra credit. A brief overview on designing integration solutions using enterprise integration patterns was given to students a week ahead of conducting the experiment. A one page summary that can be found in the Appendix D was also supplied to students during data collection. At the start of data collection, subjects were also provided with a demonstration of how to use the research prototypes. The experiment was conducted mid-term, in the eighth week of a 14-week semester, so that students had the required skills to understand and select the appropriate patterns to design integration solutions. The experiment was conducted as an in-class assignment where students use their entire class time to complete the experiment (one hour and fifteen minutes—the first fifteen minutes were used for distributing materials and demonstrating the research prototypes).
The final experiment was conducted as a crossover design (AB-BA) (Kuehl, 2000) by randomizing the assignment of problem levels and research prototype versions. Therefore each subject was arbitrarily assigned to one of the four groups listed in table 6.4. Out of 69 participants, 16 did not successfully complete every task. Of those participants that did successfully complete all tasks, 13 each from Group 1, 2, and 3, and remaining 14 from Group 4. In order to have a balanced design only the first 13 responses from each group were used for data analysis.

<table>
<thead>
<tr>
<th>Assignment Group</th>
<th>Group Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1 – CSnoDS</td>
<td>Sequence: (1) Complex problem using research prototype WITHOUT LAP constructs (2) Simple problem using research prototype WITH LAP constructs</td>
</tr>
<tr>
<td>Group 2 – CSwithDS</td>
<td>Sequence: (1) Complex problem using research prototype WITH LAP constructs (2) Simple problem using research prototype WITHOUT LAP constructs</td>
</tr>
<tr>
<td>Group 3 – SCnoDS</td>
<td>Sequence: (1) Simple problem using research prototype WITHOUT LAP constructs (2) Complex problem using research prototype WITH LAP constructs</td>
</tr>
<tr>
<td>Group 4 – SCwithDS</td>
<td>Sequence: (1) Simple problem using research prototype WITH LAP constructs (2) Complex problem using research prototype WITHOUT LAP constructs</td>
</tr>
</tbody>
</table>

6.8 Data collection

The solutions developed by the subjects were evaluated in terms of the effectiveness of the solution using the following variables: Type I (omission) errors, Type II (commission) errors, Type III (consistency) errors, the time taken for solution development, and the number of steps taken for solution development. Data on variables such as Type I, II, and III errors will be gathered by evaluating the solutions developed by subjects against the ideal solutions developed
by the author. Appendix E provides the ideal solutions for both the simple and complex problems used in the experiment. Appendix E also includes a classification of error types for patterns selected by subjects against the given ideal pattern. Data on variables such as the time and number of steps taken for developing solutions were obtained from the log files of the research prototype that records all activities performed by the subjects.

6.9 Analysis

To test the four research hypotheses, a Multivariate Analysis of Variance (MANOVA) (Tabachnick & Fidell, 2001) was used first and then followed up with an Analysis of Variance (ANOVA) for each dependent variables (Tabachnick & Fidell, 2001). The purpose of this experiment is to identify whether the independent variables (design support levels and problem complexity levels) have any significant effect on the dependent variables (Type I, II, and III design errors; design effort; number of tasks; and time taken) that measure the effectiveness of the design support with LAP constructs. The MANOVA technique can be employed when there is correlation among dependent variables and the researcher is seeking a single, overall statistical test instead of performing multiple individual tests (Tabachnick & Fidell, 2001). Statistical software Statistical Package for the Social Sciences (SPSS) (SPSS, 2006) was used to perform the data analysis.

6.9.1 Assumptions of MANOVA

In order for MANOVA analysis to be valid, the following assumptions must be met. The first and most basic assumption is that all observations need to be independent of one another (Tabachnick & Fidell, 2001). Each subject designs integration solutions independently for two
different problems using two different research prototypes; therefore, the independence assumption is satisfied. The second assumption is multivariate normality, or that all dependent variables and their linear combinations are normally distributed (Tabachnick & Fidell, 2001). Univariate normality for each variable is a necessity for multivariate normality, therefore, as a first step univariate normality of each variable is assessed (DeCarlo, 1997). Univariate normality of variables does not ensure multivariate normality; however, univariate normality does increase the likelihood of multivariate normality (Tabachnick & Fidell, 2001). Departures from normality can be detected using a combination of skewness and kurtosis coefficients as well as by assessing normal probability plots (Stevens, 2002). If the absolute ratios of skewness and kurtosis to their respective standard errors are less than 2, then the assumption of normality is not rejected (Leech, Barrett, & Morgan, 2007). Table 6.5 provides a descriptive statistical summary of all the variables along with their skewness and kurtosis coefficients.

<table>
<thead>
<tr>
<th>Table 6.5</th>
<th>Descriptive statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sample N</strong></td>
<td><strong>Mean</strong></td>
</tr>
<tr>
<td>Independent variables</td>
<td></td>
</tr>
<tr>
<td>Design support</td>
<td>104</td>
</tr>
<tr>
<td>Problem complexity</td>
<td>104</td>
</tr>
<tr>
<td>Dependent variables</td>
<td></td>
</tr>
<tr>
<td>Type I errors</td>
<td>104</td>
</tr>
<tr>
<td>Type II errors</td>
<td>104</td>
</tr>
<tr>
<td>Type III errors</td>
<td>104</td>
</tr>
<tr>
<td>Number of tasks</td>
<td>104</td>
</tr>
<tr>
<td>Time Taken</td>
<td>104</td>
</tr>
</tbody>
</table>

The observation of normal probability plots and the combination of skewness and kurtosis coefficients indicate that not all the dependent variables follow normal distributions. To attempt to correct the non-normality of the dependent variables, Box-Cox transformations were
performed. For each variable, the lowest Mean Square Error (MSE) and the corresponding appropriate transformation parameter ($\lambda$) was obtained by running the Box-Cox SPSS syntax (BoxCoxSPSS, 1998). Each variable was transformed according to the Box-Cox transformation equation given below.

$$y^\lambda = \begin{cases} 
\frac{y^\lambda - 1}{\lambda}, & \text{when } \lambda \neq 0 \\
\log y, & \text{when } \lambda = 0 
\end{cases}$$

A detailed discussion on the Box-Cox transformation method can be found in Montgomery (Montgomery, 2001). Table 6.6 provides descriptive statistics for Box-Cox transformed variables along with the transformation parameter ($\lambda$). However, the Box-Cox transformed variables had high bivariate correlations (> 0.6) between variables Type I errors and Type III errors, between variables Type I errors and Number of Tasks, and between variables Number of Tasks and Time Taken. Additionally, Box-Cox transformed variables failed Box’s test of equality of covariance matrices, and Type II errors and Type III errors variables failed Levene’s test of equality of error variances for one-way MANOVA with Design Support as independent variable.

**Table 6.6**  Descriptive statistics for Box-Cox transformed dependent variables

<table>
<thead>
<tr>
<th></th>
<th>Lambda ($\lambda$)</th>
<th>Sample N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type I errors</td>
<td>0.50</td>
<td>104</td>
<td>6.262</td>
<td>3.740</td>
<td>0.077</td>
<td>0.237</td>
</tr>
<tr>
<td>Type II errors</td>
<td>0.10</td>
<td>104</td>
<td>2.370</td>
<td>1.070</td>
<td>0.174</td>
<td>0.237</td>
</tr>
<tr>
<td>Type III errors</td>
<td>-0.10</td>
<td>104</td>
<td>1.214</td>
<td>0.791</td>
<td>0.008</td>
<td>0.237</td>
</tr>
<tr>
<td>Number of tasks</td>
<td>0.10</td>
<td>104</td>
<td>5.368</td>
<td>0.949</td>
<td>-0.155</td>
<td>0.237</td>
</tr>
<tr>
<td>Time Taken</td>
<td>0.10</td>
<td>104</td>
<td>8.900</td>
<td>1.333</td>
<td>-0.045</td>
<td>0.237</td>
</tr>
</tbody>
</table>

In order to overcome above described MANOVA violations, logarithmic transformations were performed on the dependent variables. For Type I errors, Type II errors, and Type III errors the variables were transformed using the formula LOG10 (Type1 +1), because the addition of 1
avoids the creation of missing values for cells with zeros. For _Number of tasks_ and _Time taken_ the variables were transformed without the addition of any values, because they did not contain data with value zero. Table 6.7 provides a descriptive statistical summary for the transformed dependent variables. The observation of normal probability plots and the combination of skewness and kurtosis coefficients for transformed variables indicate that they nearly follow normal distributions.

<table>
<thead>
<tr>
<th>Table 6.7</th>
<th>Descriptive statistics for LOG transformed dependent variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Type I errors</td>
<td>104</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>104</td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>104</td>
</tr>
<tr>
<td>Log Number of tasks</td>
<td>104</td>
</tr>
<tr>
<td>Log Time Taken</td>
<td>104</td>
</tr>
</tbody>
</table>

Multivariate normality test was performed for linear combinations of logarithmic transformed variables using DeCarlo’s multinorm SPSS macro (DeCarlo, 1997; MVNSPSSmacro, 1997). The plot of ordered squared Mahalanobis distances indicates that there are mild departures from multivariate normality as some points lie away from the diagonal (DeCarlo, 1997). Given the large sample size of 104 including 26 cases for each group of the 2 x 2 design (more than recommended 20) and univariate normality exists for all dependent variables, departures from multivariate normality are not expected to cause any substantive difference in the MANOVA analysis (Tabachnick & Fidell, 2001).
6.9.2 Canonical correlation test

Canonical correlation was performed to assess the relationships between the set of dependent variables and the set of independent variables. The dependent variable set includes Log Type I errors, Log Type II errors, Log Type III errors, Log Number of tasks, and Log Time taken variables. The independent variable set includes Design Support and Problem Complexity. The assumptions of canonical correlation include linearity, multivariate normality, and homoscedasticity that were checked using a matrix scatter plot of canonical variate scores generated by a canonical correlation test. The visual inspection of scatter plots yielded no evidence of patterns therefore the data appears to meet the assumptions.
Canonical correlation extracted two pairs of canonical variates with the following statistics: both canonical correlations included Chi-SQ (10) = 196.148 and p = 0.000, and with the first removed, Chi-SQ (4) = 29.983 and p = 0.000. Thus both pairs of canonical variates accounted for significant relationships between the dependent and independent variables. Table 6.8 includes correlations and canonical coefficients for both pairs.

Table 6.8    Canonical correlation test results

<table>
<thead>
<tr>
<th>Dependent variables</th>
<th>First canonical correlation</th>
<th>Second canonical correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correlation</td>
<td>Coefficient</td>
</tr>
<tr>
<td>Log Type I errors</td>
<td>0.918</td>
<td>0.564</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>0.287</td>
<td>0.003</td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>0.769</td>
<td>0.366</td>
</tr>
<tr>
<td>Log Number of tasks</td>
<td>0.742</td>
<td>0.305</td>
</tr>
<tr>
<td>Log Time Taken</td>
<td>0.450</td>
<td>-0.060</td>
</tr>
<tr>
<td>Percent of Variance</td>
<td>0.454</td>
<td></td>
</tr>
<tr>
<td>Redundancy</td>
<td>0.369</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Independent variables</th>
<th>First canonical correlation</th>
<th>Second canonical correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correlation</td>
<td>Coefficient</td>
</tr>
<tr>
<td>Design Support</td>
<td>-0.215</td>
<td>-0.215</td>
</tr>
<tr>
<td>Problem Complexity</td>
<td>0.977</td>
<td>0.977</td>
</tr>
<tr>
<td>Percent of Variance</td>
<td>0.500</td>
<td></td>
</tr>
<tr>
<td>Redundancy</td>
<td>0.407</td>
<td></td>
</tr>
<tr>
<td>Canonical correlation</td>
<td>0.902</td>
<td></td>
</tr>
</tbody>
</table>

The first canonical correlation was 0.902, representing an 81% overlapping variance for the first pair of canonical variates, thus indicating a substantial relationship between linear combinations of dependent variables and independent variables. The patterns of variables highly correlated with a variate were interpreted with a cutoff correlation of 0.6. An examination of correlations for the first canonical correlation indicates an association between Type I errors, Type III errors, the number of tasks performed, and the problem complexity levels. The second canonical correlation was 0.511, representing a 26% overlapping variance for the second pair of canonical variates. The examination of correlations for the second canonical correlation indicates an inverse association between Type II errors and the design support levels.
6.9.3 Bivariate correlation test

A MANOVA analysis can be performed when the dependent variables are correlated. Correlation among dependent variables can be tested through a bivariate correlation test. MANOVA works well when correlations among dependent variables are at moderate levels (>0.1 and <0.6) (Leech et al., 2007). Table 6.9 shows the result of the bivariate correlation test conducted among the transformed dependent variables.

**Table 6.9 Correlation matrix among dependent variables**

<table>
<thead>
<tr>
<th></th>
<th>Log Type I errors</th>
<th>Log Type II errors</th>
<th>Log Type III errors</th>
<th>Log Number of tasks</th>
<th>Log Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Type I errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pearson Correlation</td>
<td>1</td>
<td>0.232 (*)</td>
<td>0.530 (**)</td>
<td>0.589 (**)</td>
<td>0.341 (**)</td>
</tr>
<tr>
<td>Sig. (2-tailed)</td>
<td>0.018</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pearson Correlation</td>
<td></td>
<td>1</td>
<td>0.119</td>
<td>0.382 (**)</td>
<td>0.104</td>
</tr>
<tr>
<td>Sig. (2-tailed)</td>
<td></td>
<td>0.231</td>
<td>0.000</td>
<td>0.293</td>
<td></td>
</tr>
<tr>
<td>Log Type III errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pearson Correlation</td>
<td></td>
<td></td>
<td>1</td>
<td>0.395 (**)</td>
<td>0.287</td>
</tr>
<tr>
<td>Sig. (2-tailed)</td>
<td></td>
<td></td>
<td>0.000</td>
<td>0.003</td>
<td></td>
</tr>
<tr>
<td>Log Number of tasks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Pearson Correlation</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0.696 (**)</td>
</tr>
<tr>
<td>Sig. (2-tailed)</td>
<td></td>
<td></td>
<td></td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>Log Time Taken</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pearson Correlation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sig. (2-tailed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legends

(*) Correlation is significant at the 0.05 level (2-tailed)
(**) Correlation is significant at the 0.01 level (2-tailed)

From the table 6.9, it can be observed that the correlation between the Log Number of tasks and Log Time taken variables is high (0.696). A high correlation between these two variables is expected, as they conceptually overlap. It is recommended to aggregate highly correlated variables into a composite variable. Therefore, Log Number of tasks and Log Time taken variables were aggregated into a composite variable (Design Effort) by standardizing their scores.
and summing the standardized scores. The descriptive statistics of the Design Effort variable are: a mean of 0.00, a standard deviation of 1.842, and a skewness of -0.318. A bivariate correlation test was performed for this new set of dependent variables that indicates the correlation among them is moderate. Results of the correlation test are shown in table 6.10.

**Table 6.10 Correlation matrix among dependent variables with composite variable**

<table>
<thead>
<tr>
<th></th>
<th>Log Type I errors</th>
<th>Log Type II errors</th>
<th>Log Type III errors</th>
<th>Design Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Type I errors</td>
<td>1</td>
<td>0.232 (*)</td>
<td>0.530 (**)</td>
<td>0.505 (**)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.017</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>1</td>
<td></td>
<td>0.119</td>
<td>0.264 (**)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.231</td>
<td>0.007</td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>1</td>
<td>0.370</td>
<td></td>
<td>0.370</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.000</td>
</tr>
<tr>
<td>Design Effort</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Legends

(*) Correlation is significant at the 0.05 level (2-tailed)

(**) Correlation is significant at the 0.01 level (2-tailed)

6.9.4 MANOVA test for hypothesis 1 and 2

The premise of research hypothesis 1 and 2 is that the research prototype with LAP constructs has a positive effect on both design errors and design effort. A single factor MANOVA was used to assess whether different versions of the research prototype have any effect on the linear combination of the design error and design effort variables. MANOVA was performed with the Design support variable as an independent variable and the Log Type I errors, Log Type II errors, Log Type III errors, and Design Effort variables as dependent variables. Table 6.11
provides a descriptive statistics summary for each dependent variable for different levels of the independent variables.

**Table 6.11  Descriptive statistics for design support levels as independent variable**

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Sample Size (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Log Type I errors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without LAP</td>
<td>1.1822</td>
<td>0.42095</td>
<td>52</td>
</tr>
<tr>
<td>with LAP</td>
<td>1.0725</td>
<td>0.47822</td>
<td>52</td>
</tr>
<tr>
<td>Total</td>
<td>1.1273</td>
<td>0.45167</td>
<td>52</td>
</tr>
<tr>
<td><strong>Log Type II errors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without LAP</td>
<td>1.0018</td>
<td>0.4522</td>
<td>52</td>
</tr>
<tr>
<td>with LAP</td>
<td>0.659</td>
<td>0.33224</td>
<td>52</td>
</tr>
<tr>
<td>Total</td>
<td>0.8304</td>
<td>0.43078</td>
<td>104</td>
</tr>
<tr>
<td><strong>Log Type III errors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without LAP</td>
<td>0.5868</td>
<td>0.55975</td>
<td>52</td>
</tr>
<tr>
<td>with LAP</td>
<td>0.5728</td>
<td>0.4278</td>
<td>52</td>
</tr>
<tr>
<td>Total</td>
<td>0.5798</td>
<td>0.39339</td>
<td>104</td>
</tr>
<tr>
<td><strong>Design Effort</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without LAP</td>
<td>0.6137</td>
<td>1.70355</td>
<td>52</td>
</tr>
<tr>
<td>with LAP</td>
<td>-0.6137</td>
<td>1.78334</td>
<td>52</td>
</tr>
<tr>
<td>Total</td>
<td>0.000</td>
<td>1.84173</td>
<td>104</td>
</tr>
</tbody>
</table>

The assumptions of multivariate normality were checked using matrix scatter plots and they showed no signs of failure. Box’s test of equality of covariance matrices (Box’s M= 13.055, F [10, 49740.239] = 1.250, p =0.253) indicated that there are no significant differences between the covariance matrices. Therefore, the assumption of homogeneity of covariance across groups is not violated. Levene’s test of equality of error variances indicated that none of dependent variables have significant differences, thus, the variances of each dependent variable were equal across groups. Therefore, the assumption of homogeneity of variances is not violated. Table 6.12 provides a summary of Levene’s test.

**Table 6.12  Levene’s test of equality of error variances with design support levels as independent variable**

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>df1</th>
<th>df2</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Type I errors</td>
<td>2.603</td>
<td>1</td>
<td>102</td>
<td>0.110</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>3.779</td>
<td>1</td>
<td>102</td>
<td>0.055</td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>3.213</td>
<td>1</td>
<td>102</td>
<td>0.076</td>
</tr>
<tr>
<td>Design Effort</td>
<td>0.378</td>
<td>1</td>
<td>102</td>
<td>0.540</td>
</tr>
</tbody>
</table>
The multivariate null hypothesis of equality of means over all groups for the design support and design error variables was rejected at the 95% confidence interval level (Wilks’ Lambda = 0.767, F[4, 99]=7.527, p = 0.000, multivariate $\eta^2 = 0.233$). The observed significance level for all four multivariate tests was 0.000. Thus, the results indicate a modest but highly statistically significant association between the design support levels (research prototype without LAP and with LAP) and the combined dependent variables. Table 6.13 provides a summary of the MANOVA test results.

Table 6.13  Effects of design support on design error and design effort

<table>
<thead>
<tr>
<th>Effect</th>
<th>Value</th>
<th>F</th>
<th>Hyp. df</th>
<th>Error df</th>
<th>Sig.</th>
<th>Partial Eta Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillai's Trace</td>
<td>0.233</td>
<td>7.527</td>
<td>4</td>
<td>99</td>
<td>0.000</td>
<td>0.233</td>
</tr>
<tr>
<td>Wilks' Lambda</td>
<td>0.767</td>
<td>7.527</td>
<td>4</td>
<td>99</td>
<td>0.000</td>
<td>0.233</td>
</tr>
<tr>
<td>Hotelling's Trace</td>
<td>0.304</td>
<td>7.527</td>
<td>4</td>
<td>99</td>
<td>0.000</td>
<td>0.233</td>
</tr>
<tr>
<td>Roy's Largest Root</td>
<td>0.304</td>
<td>7.527</td>
<td>4</td>
<td>99</td>
<td>0.000</td>
<td>0.233</td>
</tr>
</tbody>
</table>

The examination of univariate ANOVAs for each dependent variable indicates that Type II (commission) errors and design effort variables contribute most to distinguishing the groups. The ANOVA null hypotheses of equal means were rejected at the 0.05 level for Type II (commission) errors (F[1, 102]=19.409, p = 0.000, partial $\eta^2 = 0.160$) and design effort (F[1, 102]=12.881, p = 0.001, partial $\eta^2 = 0.112$). While Type I (omission) errors and Type III (inconsistency) errors did not contributed to distinguishing design support groups, because the univariate ANOVA indicates that the two variables do not have statistically significant differences of means. Table 6.14 provides a summary of the univariate ANOVA tests for each dependent variable.
Table 6.14  Univariate test for dependent variables with design support levels as independent variable

<table>
<thead>
<tr>
<th>Dependent Variable</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
<th>Partial Eta Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Type I errors</td>
<td>Contrast Error</td>
<td>1</td>
<td>0.313</td>
<td>1.541</td>
<td>0.217</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>102</td>
<td>0.203</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>Contrast Error</td>
<td>1</td>
<td>3.056</td>
<td>19.409</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>102</td>
<td>0.157</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>Contrast Error</td>
<td>1</td>
<td>0.005</td>
<td>0.033</td>
<td>0.856</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>102</td>
<td>0.156</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Effort</td>
<td>Contrast Error</td>
<td>1</td>
<td>39.172</td>
<td>12.881</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>Error</td>
<td>102</td>
<td>3.041</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The examination of the pairwise comparison of means indicates that the research prototype with LAP constructs performed better than the research prototype without LAP constructs. With respect to Type I errors, the mean difference for the research prototype without LAP constructs (mean = 1.182) and the research prototype with LAP constructs (mean = 1.073) is 0.110. With respect to Type II errors, the mean difference for the research prototype without LAP constructs (mean = 1.002) and the research prototype with LAP constructs (mean = 0.659) is 0.343. With respect to Type III errors, the mean difference for the research prototype without LAP constructs (mean = 0.587) and the research prototype with LAP constructs (mean = 0.573) is -0.014. With respect to design effort, the mean difference for the research prototype without LAP constructs (mean = 0.614) and the research prototype with LAP constructs (mean = -0.614) is 1.227. Table 6.15 provides a summary of comparison of means for dependent variables. Even though all the variables had positive mean differences, only Log Type II errors and Design Effort have significant differences.

From the Table 6.15, it can be observed that 95% confidence interval range for mean difference between research prototypes without LAP constructs and with LAP constructs range for Type II
errors extend from 0.188 to 0.497. Because zero is not included in this interval, the data support the hypothesis that the research prototype with LAP constructs produces less design errors than the research prototype without LAP constructs at 5% level of significance. The comparison of means test indicates a substantial relationship between design support groups and Type II errors variable. For Design Effort, 95% confidence interval ranges from 0.549 to 1.906. Because zero is not included in this interval, the data support the hypothesis that the research prototype with LAP constructs requires less design effort than the research prototype without LAP constructs at 5% level of significance. The comparison of means test indicates a moderate relationship between design support groups and Design Effort variable. Thus, the research prototype with LAP constructs produced significantly less Type II errors and required significantly less design effort than the research prototype without LAP constructs.

Table 6.15  Comparison of means for dependent variables with design support levels as independent variable

<table>
<thead>
<tr>
<th>Dependent Variable</th>
<th>Mean difference (A)</th>
<th>Std. Error</th>
<th>Sig.</th>
<th>95 % CI for Difference</th>
<th>Pooled Std. Deviation</th>
<th>Effect Size (d)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lower Bound</td>
<td>Upper Bound</td>
<td></td>
</tr>
<tr>
<td>Log Type I errors</td>
<td>0.110</td>
<td>0.088</td>
<td>0.217</td>
<td>-0.066</td>
<td>0.285</td>
<td>0.450</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>0.343</td>
<td>0.078</td>
<td>0.000</td>
<td>0.188</td>
<td>0.497</td>
<td>0.392</td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>0.014</td>
<td>0.078</td>
<td>0.856</td>
<td>-0.14</td>
<td>0.168</td>
<td>0.394</td>
</tr>
<tr>
<td>Design Effort</td>
<td>1.227</td>
<td>0.342</td>
<td>0.001</td>
<td>0.549</td>
<td>1.906</td>
<td>1.743</td>
</tr>
</tbody>
</table>

Legend: (A) - Mean difference between Design Support WITHOUT LAP and WITH LAP

(d) Effect size d = A/ Pooled SD, where in Pooled SD is average of SD for two groups

The results presented above clearly support both the research hypotheses 1 and 2. MANOVA analysis suggests that there is significant difference between the research prototype levels for the linear combination of design error and design effort variables. Univariate ANOVA analysis suggests that the Type II errors and design effort variables are significantly different. The further examination of means for both Type II errors and design effort variables indicates that the
research prototype with LAP constructs produces less design errors and requires less design effort compared to the research prototype without LAP constructs.

6.9.5 MANOVA for hypothesis 3 and 4

The premise of the research hypotheses 3 and 4 is that the research prototype with LAP constructs performs better for complex problems than for simple problems in comparison to the research prototype without LAP constructs. To test hypotheses 3 and 4, a two-factor MANOVA was used. This MANOVA was performed with the Design support and Problem complexity variables as independent variables and the Log Type I errors, Log Type II errors, Log Type III errors, and Design Effort variables as dependent variables. Table 6.16 provides a summary of descriptive statistics for each dependent variable for the different levels of independent variables.

Table 6.16 Descriptive statistics for design support and problem complexity levels as independent variable

<table>
<thead>
<tr>
<th></th>
<th>Design Support</th>
<th>Problem</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Sample Size (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Type I errors</td>
<td>without LAP</td>
<td>Simple</td>
<td>0.873</td>
<td>0.204</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complex</td>
<td>0.644</td>
<td>0.278</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>0.758</td>
<td>0.267</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>with LAP</td>
<td>Simple</td>
<td>1.492</td>
<td>0.347</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complex</td>
<td>1.501</td>
<td>0.085</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>1.497</td>
<td>0.250</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>Simple</td>
<td>1.182</td>
<td>0.421</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complex</td>
<td>1.073</td>
<td>0.478</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>1.127</td>
<td>0.452</td>
<td>104</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>without LAP</td>
<td>Simple</td>
<td>0.866</td>
<td>0.457</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complex</td>
<td>0.643</td>
<td>0.301</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>0.754</td>
<td>0.399</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>with LAP</td>
<td>Simple</td>
<td>1.138</td>
<td>0.412</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complex</td>
<td>0.675</td>
<td>0.366</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>0.907</td>
<td>0.451</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>Simple</td>
<td>1.002</td>
<td>0.452</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complex</td>
<td>0.659</td>
<td>0.332</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>0.830</td>
<td>0.431</td>
<td>104</td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>without LAP</td>
<td>Simple</td>
<td>0.346</td>
<td>0.198</td>
<td>26</td>
</tr>
<tr>
<td>Design Support</td>
<td>Problem</td>
<td>Mean</td>
<td>Std. Deviation</td>
<td>Sample Size (N)</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>------</td>
<td>---------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>0.261</td>
<td>0.283</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>0.304</td>
<td>0.246</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>with LAP</td>
<td>Simple</td>
<td>0.828</td>
<td>0.322</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>0.884</td>
<td>0.302</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>0.856</td>
<td>0.311</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>Simple</td>
<td>0.587</td>
<td>0.360</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>0.573</td>
<td>0.428</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>0.580</td>
<td>0.393</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>Design Effort</td>
<td>without LAP</td>
<td>Simple</td>
<td>-0.132</td>
<td>1.581</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>-1.789</td>
<td>1.536</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>-0.961</td>
<td>1.755</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>with LAP</td>
<td>Simple</td>
<td>1.360</td>
<td>1.505</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>0.561</td>
<td>1.121</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>0.961</td>
<td>1.374</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>Simple</td>
<td>0.614</td>
<td>1.704</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>-0.614</td>
<td>1.783</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>0.000</td>
<td>1.842</td>
<td>104</td>
<td></td>
</tr>
</tbody>
</table>

Box’s test of the equality of covariance matrices (Box’s M= 94.429, F [30, 27494.033] = 2.919, p <0.5) indicates that there are significant differences between the covariance matrices. Therefore, the assumption of homogeneity of covariance across groups is violated. Levene’s test of the equality of error variances indicates that the Log Type I errors variable has significant differences, therefore, the assumption of homogeneity of variances is also violated. However, MANOVA is robust regarding violations of the assumption of homogeneity of variance and covariance when the sample sizes are equal for all factors (Tabachnick & Fidell, 2001). Because the groups are equal in size, the test should not be strongly affected by this violation. Table 6.17 provides summary of Levene’s test.
Table 6.17  Levene's test of equality of error variances with design support and problem complexity levels as independent variable

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>df1</th>
<th>df2</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Type I errors</td>
<td>2.979</td>
<td>3</td>
<td>100</td>
<td>0.035</td>
</tr>
<tr>
<td>Log Type II errors</td>
<td>1.204</td>
<td>3</td>
<td>100</td>
<td>0.312</td>
</tr>
<tr>
<td>Log Type III errors</td>
<td>2.069</td>
<td>3</td>
<td>100</td>
<td>0.109</td>
</tr>
<tr>
<td>Design Effort</td>
<td>1.053</td>
<td>3</td>
<td>100</td>
<td>0.373</td>
</tr>
</tbody>
</table>

The multivariate test indicates that the main effects as well as the interaction between the Design support and Problem complexity groups are significant. The main effect for the Design Support variable was found significant at the 95% confidence interval level (Wilks’ Lambda = 0.719, F [4, 97] = 9.498, p = 0.000, multivariate η² = 0.281). The main effect for the Problem Complexity variable was found significant at the 95% confidence interval level (Wilks’ Lambda = 0.187, F [4, 97] = 105.407, p = 0.000, multivariate η² = 0.813). The interaction between the Design support and Problem complexity variables was found significant at the 95% confidence interval level (Wilks’ Lambda = 0.869, F (4, 97) = 3.654, p = 0.008, multivariate η² = 0.131). The observed significance level for all four multivariate tests for the main effects was 0.000 and for the interaction was 0.008. Table 6.18 provides a summary of MANOVA test results.

Table 6.18  Effects of design support and problem complexity on design error and design effort

<table>
<thead>
<tr>
<th>Effect</th>
<th>Pillai's Trace</th>
<th>Value</th>
<th>F</th>
<th>Hyp. df</th>
<th>Error df</th>
<th>Sig.</th>
<th>Partial Eta Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Support</td>
<td>Wilks' Lambda</td>
<td>0.281</td>
<td>9.498</td>
<td>4</td>
<td>97</td>
<td>0.000</td>
<td>0.281</td>
</tr>
<tr>
<td></td>
<td>Hotelling's Trace</td>
<td>0.392</td>
<td>9.498</td>
<td>4</td>
<td>97</td>
<td>0.000</td>
<td>0.281</td>
</tr>
<tr>
<td></td>
<td>Roy's Largest Root</td>
<td>0.392</td>
<td>9.498</td>
<td>4</td>
<td>97</td>
<td>0.000</td>
<td>0.281</td>
</tr>
<tr>
<td>Problem Complexity</td>
<td>Wilks' Lambda</td>
<td>0.813</td>
<td>105.407</td>
<td>4</td>
<td>97</td>
<td>0.000</td>
<td>0.813</td>
</tr>
<tr>
<td></td>
<td>Hotelling's Trace</td>
<td>4.347</td>
<td>105.407</td>
<td>4</td>
<td>97</td>
<td>0.000</td>
<td>0.813</td>
</tr>
<tr>
<td></td>
<td>Roy's Largest Root</td>
<td>4.347</td>
<td>105.407</td>
<td>4</td>
<td>97</td>
<td>0.000</td>
<td>0.813</td>
</tr>
<tr>
<td>Design Support *</td>
<td>Wilks' Lambda</td>
<td>0.131</td>
<td>3.654</td>
<td>4</td>
<td>97</td>
<td>0.008</td>
<td>0.131</td>
</tr>
<tr>
<td>Problem Complexity</td>
<td>Hotelling's Trace</td>
<td>0.151</td>
<td>3.654</td>
<td>4</td>
<td>97</td>
<td>0.008</td>
<td>0.131</td>
</tr>
<tr>
<td></td>
<td>Roy's Largest Root</td>
<td>0.151</td>
<td>3.654</td>
<td>4</td>
<td>97</td>
<td>0.008</td>
<td>0.131</td>
</tr>
</tbody>
</table>
Given that the interaction between the Design support and Problem complexity variables is significant, a follow up univariate ANOVA was performed for each dependent variable with the Design support and Problem complexity variables as fixed factors. Table 6.19 provides a summary of the univariate ANOVA tests for each dependent variable. Univariate ANOVA indicated that for the interactions between Design support and Problem complexity only the Log Type I errors variable has significant difference. However, the Log Type I errors variable failed the Levene’s Test of equality, violating the assumption of homogeneity of variances. Therefore, I refrain from making any statistical conclusion on the Type I error variable. Because the other dependent variables are not found significant for interactions between the Design support and Problem complexity variables, it is concluded that the effects of the Design support group (with LAP constructs and without LAP constructs) on design error and design effect is not dependent on the Problem complexity groups (simple and complex). Therefore, research hypotheses 3 and 4 are not supported. Thus, research hypotheses 3 and 4 cannot be statistically validated for the integration problems with 5 and 16 interactions.

### Table 6.19  Univariate test for dependent variables with design support and problem complexity levels as independent variable

<table>
<thead>
<tr>
<th>Independent Variable</th>
<th>Dependent Variable</th>
<th>Mean Square</th>
<th>df</th>
<th>F</th>
<th>Sig.</th>
<th>Partial Eta Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Support</td>
<td>Log Type I errors</td>
<td>0.313</td>
<td>1</td>
<td>5.076</td>
<td>0.026</td>
<td>0.048</td>
</tr>
<tr>
<td></td>
<td>Log Type II errors</td>
<td>3.056</td>
<td>1</td>
<td>20.264</td>
<td>0.000</td>
<td>0.168</td>
</tr>
<tr>
<td></td>
<td>Log Type III errors</td>
<td>0.005</td>
<td>1</td>
<td>0.065</td>
<td>0.799</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>Design Effort</td>
<td>39.172</td>
<td>1</td>
<td>18.7</td>
<td>0.000</td>
<td>0.158</td>
</tr>
<tr>
<td>Problem Complexity</td>
<td>Log Type I errors</td>
<td>14.171</td>
<td>1</td>
<td>229.982</td>
<td>0.000</td>
<td>0.697</td>
</tr>
<tr>
<td></td>
<td>Log Type II errors</td>
<td>0.602</td>
<td>1</td>
<td>3.994</td>
<td>0.048</td>
<td>0.038</td>
</tr>
<tr>
<td></td>
<td>Log Type III errors</td>
<td>7.943</td>
<td>1</td>
<td>101.01</td>
<td>0.000</td>
<td>0.503</td>
</tr>
<tr>
<td></td>
<td>Design Effort</td>
<td>95.937</td>
<td>1</td>
<td>45.798</td>
<td>0.000</td>
<td>0.314</td>
</tr>
<tr>
<td>Design Support *</td>
<td>Log Type I errors</td>
<td>0.367</td>
<td>1</td>
<td>5.961</td>
<td>0.016</td>
<td>0.056</td>
</tr>
<tr>
<td>Problem Complexity</td>
<td>Log Type II errors</td>
<td>0.377</td>
<td>1</td>
<td>2.501</td>
<td>0.117</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>Log Type III errors</td>
<td>0.128</td>
<td>1</td>
<td>1.631</td>
<td>0.204</td>
<td>0.016</td>
</tr>
<tr>
<td></td>
<td>Design Effort</td>
<td>4.789</td>
<td>1</td>
<td>2.286</td>
<td>0.134</td>
<td>0.022</td>
</tr>
</tbody>
</table>
6.10 Discussion

The aim of this experiment is to assess the appropriateness and utility of the LAP constructs embedded in the design support for accessing and selecting appropriate integration patterns for designing integration solutions. To facilitate this assessment of LAP constructs two different research prototypes were developed. One prototype embeds LAP constructs to suggest an appropriate list of integration patterns for a given set of requirement (the research prototype with LAP constructs) and the other prototype provides a complete list of integration patterns for any given set of requirements (the research prototype without LAP constructs). The two prototypes were tested based on two different problem complexities, a simple problem with 5 interactions and a complex problem with 16 interactions.

The experiments were conducted in an advanced integration course using students for the subject pool. Subjects were asked to design integration solutions for a given pair of problem sets and research prototypes sets. The solutions developed by subjects were evaluated based on the ideal solutions to determine the number of design errors in each subject’s solution. Log files generated by the research prototype were used to identify the number of tasks and time taken to design solutions. MANOVA analysis was performed on the data gathered to test all four research hypothesis. Table 6.20 lists the research hypotheses and gives a summary of data analysis results.
### Table 6.20  Research hypothesis results summary

<table>
<thead>
<tr>
<th>Hypotheses</th>
<th>Test Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1: Designs produced with the support of LAP constructs contain fewer design errors than designs produced without the support of LAP constructs.</td>
<td>Single factor MANOVA with design support groups as independent variable and then followed up with ANOVA to identify significant dependent variables were significant contributors.</td>
<td>Supported Type II (commission) errors was the significant contributor.</td>
</tr>
<tr>
<td>H2: Designs produced with the support of LAP constructs require less design effort than designs produced without the support of LAP constructs.</td>
<td>Single factor MANOVA with design support groups as independent variable and then followed up with ANOVA to identify significant dependent variables were significant contributors.</td>
<td>Supported Design effort was the significant contributor.</td>
</tr>
<tr>
<td>H3: The use of design support with LAP constructs lead to greater reduction in design errors for complex integration problems than for simple integration problems.</td>
<td>Two factor MANOVA with design support and problem complexity groups as independent variables and then followed up with ANOVA to identify significant dependent variables were significant contributors.</td>
<td>Not Supported MANOVA test indicated significance for interactions between design support and problem complexity groups. However, ANOVA indicated that none of the dependent variables were significant for interaction between these groups.</td>
</tr>
<tr>
<td>H4: The use of design support with LAP constructs lead to greater reduction in design effort for complex integration problems than for simple integration problems.</td>
<td>Two factor MANOVA with design support and problem complexity groups as independent variables and then followed up with ANOVA to identify significant dependent variables were significant contributors.</td>
<td>Not Supported MANOVA test indicated significance for interactions between design support and problem complexity groups. However, ANOVA indicated that none of the dependent variables were significant for interaction between these groups.</td>
</tr>
</tbody>
</table>

The results of the MANOVA analyses presented in the previous section indicate that the research prototype with LAP constructs produces quality integration solutions with less effort required by the designers than the research prototype without LAP constructs. This study shows that the usage of LAP constructs is effective at reducing design errors, and thereby, produces quality solutions. This study also shows that the usage of LAP constructs is effective in reducing the lead time needed to produce solutions while still maintaining the quality of those solutions. Therefore, the usage of LAP constructs, such as speech acts, for the selection of appropriate integration patterns based on integration requirements is validated.
The MANOVA analysis of the effects of design support levels indicated that there is a significant difference between the research prototypes. The follow up ANOVA on each dependent variable indicates that the Type II (commission) errors and design effort were the significant contributors to the difference between the research prototypes; while Type I (omission) errors and Type III (inconsistency) errors were not significant. The significance of Type II errors indicates that the research prototype with LAP constructs is effective at reducing errors caused by the selection of patterns that do not satisfy the integration requirements. This shows that the LAP constructs are effective in providing lists of patterns that are appropriate for a given set of requirements. The significance of the design effort ratio indicates that the research prototype with LAP constructs is effective in reducing the number of steps and time required to develop solutions. This shows that LAP constructs are instrumental in reducing the lead time for designing integration solutions while also producing higher quality solutions.

The insignificance of Type I errors indicates that the research prototype with LAP constructs is not effective at reducing errors caused by no pattern selections or by the selection of patterns that provide a partial satisfaction of requirements. The research prototype provides a tabular summary of patterns selected for each requirement in the same window where the designer can browse and make pattern selections. When no pattern is selected for a requirement, the summary table indicates to the designer that no selection has been made. The summary table is automatically updated when the designer selects or deselects a pattern for a requirement. Therefore, the research prototype does include appropriate measures to indicate to designers that no pattern selection is made. However, the research prototype did not include any measures to indicate the partial satisfaction of requirements. The insignificance of Type III errors indicates
that the research prototype with LAP constructs is not effective at reducing errors caused by the selection of conflicting patterns and/or inferior patterns. The research prototype did not include any measures to indicate conflicts among the selected patterns or to provide comparisons of one pattern against other related patterns. The insignificance of Type I and III errors indicates that the research prototype should include features to identify and warn designers about partial solution sets as well as conflict resolutions, and to provide comparisons of related patterns.

The MANOVA analysis for the effects of the design support and problem complexity levels indicates that there is a significant difference in the interaction between design support groups and problem complexity groups. However, the univariate ANOVA analysis indicates that there is no significant difference for either of the dependent variables regarding the interaction between design support groups and problem complexity groups. This insignificance of interaction indicates that there is no significant difference in the performance of research prototypes when the problem complexity increases from 5 interactions to 16 interactions. This insignificance could be influenced by other plausible factors such as (a) the need for integration problem complexity levels with interaction differences greater than 11 and (b) the learning that may occur during the sequential process of selecting patterns.
7.1 Synopsis

The objective of this dissertation is to stimulate a discourse on and a search for the appropriate theoretical foundations for web services. Current web service efforts are confronted with several complexities that demand a theoretical foundation that can provide adequate guidance and accelerate web service efforts (Umapathy & Purao, 2007b). In this dissertation I argue that the Language-Action Perspective (LAP) is an appropriate theoretical foundation for web services because there is a close match between the core principles of web services and those of the LAP approach. Therefore, this dissertation is aimed at validating the appropriateness of the LAP approach for web services through three inter-dependent essays.

The first essay is conducted at the architecture level to investigate the appropriateness of LAP to describe and explain web services architecture. The essence of this essay is a reference framework (Umapathy & Purao, 2007a) developed based on the LAP constructs that identifies the specific functions that need to be addressed within the interoperable web services architecture and an assessment of the existing three standard stacks. The reference framework provides guidance to all web service stakeholders to make sense of web services architecture. This essay uses conceptual development and logical argument for a research method. One of the findings from the first essay indicates that there is a lack of standards and mechanisms to develop conversation specifications (cpXML, 2002) that guide the interactions among web services.

The second essay builds on the findings from the first essay. This essay is conducted at the conceptual level to demonstrate the appropriateness of using LAP constructs to develop web service solutions. The domain used for this investigation is enterprise integration (Jinyoul Lee et
This essay uses design science research (Hevner et al., 2004) for a research methodology. The main purpose of this essay is to construct a software artifact that assists designers in the development of web service based integration solutions. The software artifact embeds a mechanism that uses LAP constructs and a corresponding knowledge base containing heuristics on the usage of integration design knowledge (or enterprise integration patterns) (Hohpe & Woolf, 2004). The software artifact semi-automates the process of selecting appropriate integration patterns as integration solutions and translating the tactics represented in the selected patterns into web service conversation specifications. Thus, the LAP-based mechanism embedded within the software artifact allows designers to develop quality web service based integration solutions while reducing the lead time required for developing them.

The third essay is conducted at the empirical level to evaluate the appropriateness of the LAP constructs used for developing the software artifact in the second essay. This essay uses a controlled experiment (Shadish et al., 2001) as its research methodology. The essence of this essay is to assess the utility of the LAP constructs embedded in the software artifact for designing integration solutions. To facilitate the assessment of LAP constructs, two different research prototypes were developed, one embedded with LAP constructs and another without LAP constructs. These two prototypes were tested based on two different problem complexities, a simple problem with 5 interactions and a complex problem with 16 interactions. The experiment was conducted using advanced integration course students for its subject pool. The subjects were asked to design integration solutions for a given pair of problem sets and research prototype sets. The solutions developed by the subjects were evaluated in terms of the kinds of design errors produced and the effort required in developing solutions. A MANOVA analysis,
along with a follow up ANOVA, was performed on data gathered to test the utility of LAP constructs in terms of design errors and design effort. This analysis revealed that the software artifact with LAP constructs produces higher quality solutions and requires less effort to develop those solutions when compared to the software artifact without LAP constructs. The analysis also indicated that the software artifact with LAP constructs is not effective in reducing design errors caused by solutions with partial and conflicting sets of selected patterns.

### 7.2 Information-Technology-People (ITP) perspective

One of the challenges for doctoral dissertation research in the College of Information Sciences and Technology (IST) is to take a thematic view and position on the Information-Technology-People (ITP) perspective. Figure 7.1 summarizes the contributions of this dissertation research from the ITP perspective. Below, I provided a discussion on the contributions to the ITP perspective of each of the three essays within this dissertation.

The first essay contributes a LAP-inspired reference framework for the web services community. This reference framework is a discrete entity that can exist on its own, because it provides guidance to the web services community for understanding and using interoperable web services architecture (Sawyer & Chen, 2002). The guidance provided by the reference framework, therefore, represents the *information* element of the ITP perspective. Web services community represents the *people* element of the ITP perspective, because the web services community is a group of individuals who are interested in using web services architecture to their benefit (Sawyer & Chen, 2002). The first essay, therefore, focuses on developing a reference framework
in the context of the web services community, thus, it contributes to the *information-people* aspect of the ITP perspective.

**Figure 7.1. Contributions of this dissertation from ITP perspective**

The *second essay* contributes a software artifact for enterprise integration designers to develop enterprise integration solutions using web services. The software artifact uses in-built algorithms to understand the business processes developed by designers to guide them with designing integration solutions. Because it has the computational capability to model, process, and develop integration solutions, the software artifact represents the *technology* element of the ITP perspective (Orlikowski & Iacono, 2001). The software artifact uses a knowledge base that contains heuristic information for identifying appropriate solutions. The software artifact also provides methodological guidance to designers for developing integration solutions using web services. Methodological guidance and heuristic information represents the *information* element of the ITP perspective, given that this guidance and information are discrete entities that can be stored and retrieved whenever required (Sawyer & Chen, 2002). The second essay, therefore, focuses on the creation of an IT artifact for designing integration solutions and it contributes to the *information-technology* aspect of the ITP perspective.
The third essay evaluates the software artifact in terms of the effectiveness of the solutions it develops using a controlled experiment. In this experiment, a panel of designers who are experts in the enterprise integration domain were asked to design enterprise integration solutions using the software artifact. The software artifact represents the technology element of the ITP perspective because the software artifact is a computational entity used for the construction of integration solutions (Orlikowski & Iacono, 2001). The panel of experts represents the people element of the ITP perspective, because they are individuals with the required skill set to design enterprise integration solutions (Sawyer & Chen, 2002). The third essay, therefore, focuses on the interactions between the designers and the software artifact, and it contributes to the technology-people aspect of the ITP perspective.

### 7.3 Contributions and outcomes

| Essay 1 | • Commonalities among web service initiatives – Facilitating communication among services  
• LAP-inspired framework for web services  
• Assessment of LAP-inspired framework against to three web service initiatives  
• Insights for refining web service initiatives standardization efforts |
| --- | --- |
| Essay 2 | • Conceptual model for accessing and selecting integration patterns  
• List of LAP constructs appropriate for enterprise integration  
• Knowledge base for enterprise integration patterns  
• Methodology and mechanism for generating web service conversation specifications as enterprise integration solutions  
• Software artifact that assists designers to develop web services-based integration solutions |
| Essay 3 | • Operationalizing the variables for assessment including measures for design errors and design effort  
• The Assessment of utility of LAP constructs for designing integration solutions  
• Interpretation of results for research and practice |
Table 7.1 provides a summary of the contributions from each of the three essays. Apart from these contributions, the greatest value of this research is that it starts a discussion on the need for a comprehensive theoretical perspective in the web services community. The web services community will benefit from the results of this research because this study provides a holistic view of the web services framework and identifies missing elements in the existing web services framework. These results provide appropriate insights for coherent web service standardization efforts. Researchers of services science\(^6\) will also be interested in the outcomes of this research, because the development of an appropriate theoretical framework that can answer the central questions of services science is one of the key challenges in this discipline (Chesbrough & Spohrer, 2006; SSME-IBMARC, 2006). Given that web services is a subset of services science, the results of this research can be extended and applied in the services science discipline.

The LAP community will benefit from this research because it contributes to the ongoing discussion on how to help the LAP approach reach mainstream computing. This research could motivate LAP researchers and provide insights that help them to make contributions in the web services domain. The LAP community will be interested in the outcomes of this research as it could subdue their anxiety over how to reach mainstream organizational computing.

7.3.1 Contributions to research

Through the assessment of the existing web service standardization efforts as opposed to the LAP-inspired framework in the first essay, this research provides insights and direction for web

\(^6\) Services science is a multi-disciplinary approach to explore the underlying phenomenon of the services economy and its implications (Chesbrough & Spohrer, 2006; IBMSummit, 2004). For more information on services science, please refer CACM July 2006 issue.
service researchers on the refinement of existing standards and the development of new ones. Through the creation of a conceptual model for accessing and selecting appropriate integration patterns and a mechanism for translating business processes into web service conversation specifications in the second essay; this research contributes to the enterprise integration domain. The software artifact is constructed and evaluated following design science research guidelines; thus, this study contributes to the field of design science research. Through the evaluation of the LAP constructs embedded in the software artifact in the third essay, this research provides demonstrates for the appropriateness of using LAP constructs to solve web service problems.

7.3.2 Contributions to practice

Through the development of the LAP-inspired framework in the first essay, this research provides a holistic view on web service architecture. The LAP-inspired framework provides guidance for web service stakeholders to understand the interoperable architecture of web services. Through the development of a software artifact that generates web service conversation specifications as enterprise integration solutions in the second essay; this research makes a practical contribution to the enterprise integration domain. Through the evaluation of the LAP constructs embedded in the software artifact in the third essay, this research exhibits the practical utility of using LAP constructs to develop web service solutions.

7.3.3 Contributions to the I-discipline

The objectives of IST are lead in the education and research necessary to develop solutions for a globalized society, and to build an intellectual community of scholars with diversified knowledge and skills. Therefore, IST should learn and teach non-traditional information systems
theories such as LAP in order to develop new solutions for globalized society. This dissertation captures and portrays the importance, relevance, and application of LAP in the web services domain. The success of this research could eventually influence IST to consider making LAP as part of its curriculum. The success of this research could also motivate IST researchers to use non-traditional multi-disciplinary theories and to develop effective solutions.

7.4 Limitations and future research opportunities

The scope of this study is limited to using LAP constructs to understand and address the problems of web services. Therefore, one future research need is a review of web service publication avenues aimed at identifying a list of theories being used in the domain. This review will extend the review of flagship conferences discussed in the chapter 1. The LAP approach could be assessed against that set of theories to identify its strength and weakness. Table 7.2 provides a summary of limitations and future research opportunities.

| First essay | Limited opportunity to test the LAP-inspired framework over time as part of a dissertation. | Empirical validation of the LAP-inspired reference framework. |
| Second essay | Software artifact focus on one process at a time instead of considering a portfolio of processes. | Extend software artifact to consider set of processes to recommend patterns and assist in designing integration solutions. |
| Third essay  | Laboratory experiment with student subjects. | Empirical evaluation of the software artifact in real world settings. |

With respect to the first essay, it provides an assessment of the LAP-inspired framework against the existing three web service initiatives; however, it does not provide an empirical validation of the framework. Thus, the empirical validation of the LAP-inspired reference framework is identified as future work. With respect to the second essay, the software artifact assists designers
to design solutions considering one process at a time, instead of considering a portfolio of processes. Thus, extending the software artifact to consider a set of processes at same time for designing web service solutions as integration solutions is identified as future work. With respect to the third essay, the software artifact with LAP constructs was evaluated in a laboratory setting using students as subjects—that approach has limitations when the results are generalized to real-world settings. Thus, an empirical evaluation of the software artifact with LAP constructs in a real-world setting with experienced designers is identified as future work.

Even though this dissertation applies the LAP approach only in the context of web services, the LAP approach can be applied in other contexts where communication is used to perform actions. For instance, the LAP approach can be applied in the context of Co-Design of business processes and information systems. Co-Design provides the ability to dynamically reflect changes in the business process level to the information systems level and vice versa (Liu, Sun, & Bennett, 2002; Tam, Choi, & Chung, 1994). Such dynamic changes would require substantial communication and collaboration among various stakeholders as well as systems (Umapathy, 2007). Given the long history of the LAP approach to support communication and collaboration among humans as well as systems, the LAP approach would be an appropriate theoretical perspective to investigate problems in the Co-Design domain. Similarly, the LAP approach specifically, Habermas’ Theory of Communicative Action (Habermas, 1984) can be used to understand the underlying process for creation of IT standard artifacts. Standard consortiums such as W3C, OASIS, IETF, and OMG create various IT standards through a process in where participants argue, collaborate, deliberate, propose, negotiate, question and support on details regarding design of components that make up the IT artifact (Cargill, 1989; Umapathy, Paul,
Purao, Bagby, & Mitra, 2007). The underlying mechanisms behind IT standardization processes that lead to creation of IT standards which we use everyday are not well understood and severely under researched (West, 2003). The LAP approach can be used to understand what kinds of communication and collaboration takes place among participants in variety of situations including IT standardization processes (de Moor & Aakhus, 2006; Heng & de Moor, 2003).

### 7.5 Conclusion

The central significance of this dissertation is that it demonstrates how LAP can be used to solve problems in the web services domain. This demonstration was performed through three different perspectives. First, from the theoretical perspective, it demonstrates how LAP can be used to understand web service architecture. Second, from the application perspective, it demonstrates how LAP can be used to solve web service domain problems. Finally, from the empirical perspective, it demonstrates that the principles underlying the LAP approach are of practical importance. I hope this study encourages web service researchers to consider using LAP, or other suitable theories, to understand and solve web service problems instead of following a piecemeal approach. I hope this study encourages LAP researchers to use web service as an application domain to help LAP reach the mainstream computing paradigm.
REFERENCES


BoxCoxSPSS. (1998). Box-Cox transformation SPSS Syntax. College Station, TX Department of Statistics, Texas A&M University


Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software: Addison Wesley.


presented at the IEEE International Enterprise Distributed Object Computing Conference (EDOC).


<table>
<thead>
<tr>
<th>Standard</th>
<th>Initiative</th>
<th>Name</th>
<th>Description of the standard</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP</td>
<td>All three</td>
<td>File Transfer Protocol</td>
<td>Specifies a protocol to share documents remotely.</td>
<td><a href="http://www.w3.org/Protocols/rfc959/">http://www.w3.org/Protocols/rfc959/</a></td>
</tr>
<tr>
<td>MOWS</td>
<td>W3C</td>
<td>Management of Web Services</td>
<td>Defines manageability of web service endpoints and resources exposed.</td>
<td><a href="http://docs.oasis-open.org/wsdm/2004/12/wsdm-mows-1.0.pdf">http://docs.oasis-open.org/wsdm/2004/12/wsdm-mows-1.0.pdf</a></td>
</tr>
<tr>
<td>OWL</td>
<td>Semantic web services</td>
<td>Ontology Web Language</td>
<td>Facilitates greater machine interpretability of web content by providing additional vocabulary along with a formal semantics.</td>
<td><a href="http://www.w3.org/2004/OWL/">http://www.w3.org/2004/OWL/</a></td>
</tr>
<tr>
<td>OWL-S (includes SWSL and SWSL-Rules)</td>
<td>Semantic web services</td>
<td>OWL-based Web Service Ontology</td>
<td>Describes service profile, service model and service grounding.</td>
<td><a href="http://www.w3.org/Submission/OWL-S/">http://www.w3.org/Submission/OWL-S/</a></td>
</tr>
<tr>
<td>SOAP</td>
<td>All three</td>
<td>Simple Object Access Protocol</td>
<td>Specifies protocol for exchanging structured information in a decentralized, distributed environment.</td>
<td><a href="http://www.w3.org/TR/soap/">http://www.w3.org/TR/soap/</a></td>
</tr>
<tr>
<td>UDDI</td>
<td>W3C</td>
<td>Universal Description,</td>
<td>Defines a set of services supporting the</td>
<td><a href="http://www.oasis-">http://www.oasis-</a></td>
</tr>
<tr>
<td>Standard</td>
<td>Initiative</td>
<td>Name</td>
<td>Description of the standard</td>
<td>URL</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>---------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>WS-Addressing</td>
<td>W3C</td>
<td>Discovery and Integration</td>
<td>Discovery and discovery of web services.</td>
<td>open.org/committees/tc_home.php?wg_abbrev=uddi-spec</td>
</tr>
<tr>
<td>WS-BPEL</td>
<td>W3C</td>
<td>Web Services Business Process Executing Language</td>
<td>Defines a set of abstract properties to facilitate end-to-end addressing of endpoints in messages.</td>
<td><a href="http://www.w3.org/TR/ws-addr-core/">http://www.w3.org/TR/ws-addr-core/</a></td>
</tr>
<tr>
<td>WS-CDL</td>
<td>W3C</td>
<td>Web Services Choreography Description Language</td>
<td>Defines a formal language to specify external observable behavior as the presence or absence of messages that are exchanged among web services.</td>
<td><a href="http://www.w3.org/TR/ws-chor-reqs/">http://www.w3.org/TR/ws-chor-reqs/</a></td>
</tr>
<tr>
<td>WSDL</td>
<td>W3C</td>
<td>Web Services Description Language</td>
<td>Defines a formal language to describe the abstract functionality as well as the concrete details of a service.</td>
<td><a href="http://www.w3.org/TR/wsd120/">http://www.w3.org/TR/wsd120/</a></td>
</tr>
<tr>
<td>WS-Topics</td>
<td>W3C</td>
<td>Web Services Topics</td>
<td>Defines topic expression dialects and metadata associated with the web services notification system.</td>
<td><a href="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-Topics-1.2-draft-01.pdf">http://docs.oasis-open.org/wsn/2004/06/wsn-WS-Topics-1.2-draft-01.pdf</a></td>
</tr>
</tbody>
</table>
Appendix B  Experimental Instructions
Welcome to the Integration Designer Assistant (IDAssist) research project. We would like to thank you for considering participating in our research.

The purpose of this research is to assess the utility of the software we have developed, known as IDAssist. This software provides required design knowledge for designing solutions to integrate software applications. We wish to explore how users, like yourself, comprehend various design strategies and develop solutions using this software.

Your role in this current project is to use the IDAssist software to design integration solutions for a given business process scenario. You will be given specific instructions for running this software and uploading process scenarios into it. You will be provided with two different problems for which you will be expected to develop integration solutions. The software will assist you in identifying requirements (i.e., interactions between performers) for which appropriate design solutions (i.e., integration patterns) need to be identified. The software will also provide you with various alternative design solutions for each requirement. Your responsibility is to read each design option and select one or more that you find most appropriate for each requirement.

Business process scenarios that will be provided to you pertain to enterprise integration problem. A brief background on enterprise integration problems has been provided as a separate document – see “Building Enterprise Integration Solutions with Patterns” document.

Please ask me if you have any questions about the research, instructions, your role, or any of the given problem scenarios found on the following pages.
Problem 1

Below, you will find the step by step instructions for designing integration solutions using the software for problem 1:

1. Log into ANGEL course management system using your Penn State account and password.

2. Access the IST 421 course contents, which you will find under “My Courses” on your ANGEL profile page.

3. Download “IDAssist Files,” which you can find under “IDAssist” folder under “Lessons” tab in ANGEL and save it in your desktop.
   (Make sure that the downloaded file is saved with .zip as the extension.)

4. Uncompress the IDAssist files. IDAssist files will be uncompressed and it contents will be placed under IDAssist folder in your desktop.

5. Open newly created IDAssist folder by double clicking it.

6. Run IDAssistnoDS software by double clicking the “IDAssistnoDS.jnlp” file in the IDAssist folder.
   (Make sure that the title bar of the software reads as “Integration Designer Assistant – IDAssistnoDS Software”. If it does not, it means you are running wrong software. If correct, then proceed with next instruction.)

7. The software will open a process scenario selection window; please select “Loan Request Process” as your process scenario and then click the “Submit” button. See figure below.

![Integration Designer Assistance - Process Selection](image)

8. The software will load the process into a process window, see figure below. You may learn more about the process by selecting each task (presented as icons in the process window) and reading the details provided in the property window to the right (see figure below).
9. Read the integration requirements (i.e., interactions between performers) by browsing identified interactions listed in the interactions listing window, see figure below. 

*(For browsing interactions, click the “Design” menu and then “Browse Interactions” menu item. Selected interactions will be highlighted in the process)*

Each interaction is considered an integration requirement. When a particular interaction is selected, set tasks in the process window which are parts of the selected interaction will be highlighted. You can read the interaction description in the scroll box at the bottom of the screen to learn more about the selected interaction. You may browse all interactions one after another to learn more about each integration requirement.
(To browse integration patterns, click the “Design” menu and then the “Browse Integration Patterns” menu item. A list of integration patterns will be shown toward the bottom of the screen)

When you select an interaction from the dropdown list in the interactions box, a list of integration patterns will be provided in the pattern list box (see figure below). When a particular pattern is selected, details will be displayed in the pattern details window.

11. Select appropriate integration patterns for each identified interaction.
You should browse all patterns listed for the selected interaction, learn about each pattern and then select one or more patterns that you find appropriate for the selected interaction. You can select a pattern for a selected interaction by selecting the check box next to the “Use this pattern for this interaction” in the pattern details window. When a pattern is selected, it will be displayed next to the selected interaction, in the design summary window (see figure below). You can select more than one pattern for an interaction, if it is a necessity.

You should continue this step, doing the same for all the interactions listed in the interactions listing window, to select appropriate patterns for each interaction. The patterns selected represent your design solutions. A summary of the patterns you choose will be displayed in the design summary window.
12. After selecting appropriate patterns for each interaction, view the design summary report. If you are happy with your pattern selections, submit the design report by clicking on the button labeled, “Yes”. See below figure. Repeat step 11 above if you wish to change the patterns in the design summary by clicking on the button labeled, “No”.

*(For viewing design summary report, click “Design” menu and then “Design Summary” menu.)*

13. Close the IDAssistnoDS software.

*(To close the software, click the “File” menu in the upper left-hand part of the screen and then click the “Exit” menu item.)*

You have completed designing solutions for problem 1. Please proceed with problem 2.
Problem 2

The step by step instructions for designing integration solutions using IDAssist software for problem 2 appear below:

1. Run IDAssistwithDS software by double clicking the “IDAssistwithDS.jnlp” file in the IDAssist folder. (Make sure that the title bar of the software reads as “Integration Designer Assistant – IDAssistwithDS Software”. If it does not, it means you are running wrong software. If correct, then proceed with next instruction.)

2. The software will open a process selection window; please select “Online Retailer Warehouse Replenishment Process” and then click the “Submit” button. See figure below.

3. The software will load the process in the process window (see figure below). You may learn more about the process by selecting each task (represented by icons in the process window) and reading the details provided in the property window to the right (see figure below).

4. Read the integration requirements by browsing identified interactions listed in the interactions listing window, see figure below.
(To browse interactions, click the “Design” menu and then “Browse Interactions” menu item. Selected interactions will be highlighted in the process)

Each interaction is considered an integration requirement. When a particular interaction is selected, set tasks in the process window which are parts of the selected interaction will be highlighted. You can read the interaction description in the scroll box at the bottom of the screen to learn more about the selected interaction. You may browse all interactions one after another to learn more about each integration requirement.

5. Browse the integration patterns.
   (For browsing integration patterns, click the “Design” menu and then the “Browse Integration Patterns” menu item. A list of integration patterns will be shown toward the bottom of the screen)

   When you select an interaction from the dropdown list in the interactions box, a list of integration patterns will be provided in the pattern list box (see figure below). When a particular pattern is selected, details will be displayed in the pattern details window.

6. Select appropriate integration patterns for each identified interaction.
   You should browse all patterns listed for the selected interaction, learn about each pattern and then select one or more patterns that you find appropriate for the selected interaction. To select a pattern for an interaction, select the check box next to the “Use this pattern for this interaction” in the pattern details window. When a pattern is selected, it will be displayed next to the selected interaction, in the design summary window (see figure below).

   You should continue this step, doing the same for all the interactions listed in the interactions listing window, to select appropriate patterns for each interaction. The patterns selected represent your design solutions. A summary of the patterns you choose will be displayed in the design summary window.
7. After selecting appropriate patterns for each interaction, view the design summary report. If you are happy with your pattern selections, submit the design report by clicking on the button labeled, “Yes”. See below figure. Repeat step 11 above if you wish to change the patterns in the design summary by clicking on the button labeled, “No”. 
(For viewing design summary report, click “Design” menu and then “Design Summary” menu.)

8. Close the IDAssistwithDS software.
(To close the software, click the “File” menu and then “Exit”.)
**Finishing Up**

1. Compress the contents of the IDAssist folder in your desktop using compression software such as WinZip. Please use “IDAssist- your PSU ID” as format for naming compressed file. If you do not have any compression software, you may download freeware software from [http://www.winzip.com/prod_down.htm](http://www.winzip.com/prod_down.htm).

2. Upload the compressed file (with .zip extension) into the drop box named “Outputs” which can be found under “IDAssist” folder under the “Lessons” tab in ANGEL 421 course page.

3. Open and complete the “questionnaire” in ANGEL located under the “IDAssist” folder in the “Lessons” tab in the IST 421 course.

Thank you very much for participating in this study. We appreciate your time and effort and contributions to our research.

If you have any questions about this research, please contact Karthikeyan (e-mail address is available below).

Sincerely,

Karthikeyan Umapathy (kumapathy@ist.psu.edu)
School of Computing
University of North Florida
and
College of Information Sciences and Technology
The Pennsylvania State University

Sandeep Purao (spurao@ist.psu.edu)
College of Information Sciences and Technology
The Pennsylvania State University
Appendix C  Questionnaire
Personal Information

1. In which age group do you belong?
   a. 17 and under
   b. 18 - 25
   c. 26 - 35
   d. 36 - 45
   e. 46 - 59
   f. 60 +

2. Please identify your gender
   a. Female
   b. Male

3. How would you characterize yourself?
   a. Freshmen
   b. Sophomore
   c. Junior
   d. Senior
   e. Super Senior
   f. Graduate
   g. Employed as professional worker or Self employed
   h. Others, please specify

Design experiences

1. Have you taken or enrolled in any courses on enterprise integration?
   a. Yes, specify course name ________________________
   b. No

2. Describe different applications and/or technologies that you have learned or utilized to integrate applications.

3. Describe your experience/ internship related to designing software applications. Describe kinds of work produced. You may list different projects/ applications that you worked on.
Designing solutions

1. What particular aspect(s) of IDAssist software did you *like*?

2. What particular aspect(s) of IDAssist software did you *dislike*?

3. How confident do you feel that you evaluated appropriateness of all patterns listed for each identified interactions?

   Not at all confident  2  3  4  5  6  7  8  9  Very confident

4. Please indicate your recommendation scale to implement the solutions you developed.

   Not at all recommended  2  3  4  5  6  7  8  9  Highly recommended

5. How confident do you feel that solutions developed is free of errors?

   Not at all confident  2  3  4  5  6  7  8  9  Very confident

6. How demanding did you find designing solutions to be?

   Not at all demanding  2  3  4  5  6  7  8  9  Very demanding

7. Is there anything else you would like to share about your reactions to the tasks or tools that you used today?
Appendix D  Building Enterprise Integration Solutions with Patterns
**Business process**
A business process specifies tasks, performers, interactions between tasks, and resources required. Here is an example for generating a quote for a customer order. It has six tasks. For each, it specifies a performer (human actors e.g. Customer or legacy applications e.g. Cost Estimator). The process captures rules such as: (a) task 2 cannot be done unless task 1 is done, and (b) tasks 3 and 4 may be done concurrently.

**Problem: Interactions among performers**
Clearly, performers of related tasks must interact with one another to complete the process. For example, Retailer (for task 2) and Warehouse A (for task 3) may need to exchange data.

**Potential help from Integration Patterns**
When such problems are repeatedly encountered, designers identify “Patterns.” For example, a one-to-many interaction with known participants (e.g. between performers for tasks 2, 3, and 4) suggests the pattern ‘publish-subscribe.’ It shows how channels between performers may be designed to deliver a copy of an event to receivers who may have subscribed to the channel. In the example, Warehouses A and B act as subscribers. The Retailer acts as the publisher. Every time the Retailer receives a new order, it publishes a notification that the registered subscribers (Warehouses A and B) receive.

**Need for design support**
It is difficult to decide which patterns to use when faced with a process. In addition, there are several such patterns for enterprise integration (Hohpe and Woolf). And that is the problem I am interested in solving.

**The integration designer’s assistant**
I am developing a tool that can help designers (a) decompose a process into sets of interactions, (b) identify appropriate patterns for each, and finally, (c) combine the selected patterns for an overall solution.
Appendix E  Enterprise Integration scenarios
Simple Problem: Loan request processing

Description
In this process, a customer goes online to the loan broker’s web site and s/he enters several pieces of information, including name, contact details, and desired loan amount. The loan broker fulfills the request by first obtaining a credit score of the customer from a credit bureau agency. Then the loan broker forwards loan request to three different banks to obtain their quotes. Once the loan broker receives quote from all three banks it selects the best quote and returns it to the customer.

Source
Adapted and modified from Enterprise Integration Patterns book. Authors of the Enterprise Integration Patterns book utilize this example to explain how to generate test message and client and how to develop system management consoles. This example was modified to expose interactions among underlying systems.
### Ideal solutions

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Interaction type</th>
<th>Initiator(s)</th>
<th>Responder(s)</th>
<th>Ideal patterns</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>One to One</td>
<td>1</td>
<td>2</td>
<td>Request Reply</td>
<td>Interaction between loan broker and credit score agency is two-way interaction.</td>
</tr>
<tr>
<td>2</td>
<td>One to One</td>
<td>2</td>
<td>3</td>
<td>Request Reply</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>One to Many</td>
<td>3</td>
<td>4, 5, 6</td>
<td>Recipient List or Publish Subscribe Channel</td>
<td>Loan broker needs to send request to three banks which can be accomplished either by creating a recipient list of banks for each request based on certain rules or by publish each request to certain banks based on some criteria’s</td>
</tr>
<tr>
<td>4</td>
<td>Many to One</td>
<td>4, 5, 6</td>
<td>7</td>
<td>Content Enricher</td>
<td>Aggregates each quotes received from three banks and selects best quote based on loan quote selection rules</td>
</tr>
<tr>
<td>5</td>
<td>One to One</td>
<td>7</td>
<td>8</td>
<td>Point to Point Channel Document Message</td>
<td>Quote evaluator prepares best quote document and sends to the loan broker who will forward it to the customer.</td>
</tr>
</tbody>
</table>
Complex Problem: Order replenish process

Description
Nile.com is an online retailer offering electronic goods to consumers. To fulfill orders Nile.com verifies whether warehouse has items ordered by the customer are in stock. If items are in stock, order is fulfilled order fulfillment process. If items are not in stock, alternatives items are provided to customer. If certain items are not stock or when an item in stock falls below a certain threshold, the warehouse must restock the item from the relevant Manufacturer’s inventory. Nile.com performs item demand forecast to calculate number of items to be ordered and identifies a Manufacturer and places the order with the Manufacturer. In order to fulfill Nile.com order a Manufacturer verifies whether ordered item is in the stock in the Manufacturer’s inventory. If ordered items are in stock, then Manufacturer ships the items to the Nile.com warehouse. If ordered items are not in stock then Manufacturer executes a production run to build the requested items to stock in the inventory.
Source
Modified and adapted from Supply Chain Management Scenario developed by the Web Services Interoperability (WS-I).

### Ideal solutions

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Interaction type</th>
<th>Initiator(s)</th>
<th>Responder(s)</th>
<th>Ideal patterns</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>One to One</td>
<td>1</td>
<td>2</td>
<td>Point-to-Point Channel Command Message</td>
<td>It is ensured that customer order is received from the Nile.com website is consumed only once, so that there is only single invocation of order fulfillment upon receipt of customer order.</td>
</tr>
<tr>
<td>2</td>
<td>One to One</td>
<td>2</td>
<td>3</td>
<td>Point-to-Point Channel Document Message</td>
<td>Customer order is send to the warehouse to verify whether order items are in stock and it is ensured that warehouse consumes this message only once.</td>
</tr>
<tr>
<td>3</td>
<td>One to Many</td>
<td>3</td>
<td>4, 5</td>
<td>Content-Based Router</td>
<td>Based on the warehouse stock check either order fulfillment process s invoked, or process to place order to stock items is invoked.</td>
</tr>
<tr>
<td>4</td>
<td>One to Many</td>
<td>4</td>
<td>6, 8</td>
<td>Content-Based Router</td>
<td>If order has not already placed then invoke demand forecaster</td>
</tr>
<tr>
<td>5</td>
<td>One to One</td>
<td>8</td>
<td>9</td>
<td>Point-to-Point Channel Document Message</td>
<td>Upon receipt of demand forecasting for the item, list of manufacturers is obtained</td>
</tr>
<tr>
<td>6</td>
<td>One to One</td>
<td>9</td>
<td>10</td>
<td>Point-to-Point Channel Document Message</td>
<td>Upon identification a manufacturer for the item, order to stock the item is placed with the manufacturer.</td>
</tr>
<tr>
<td>7</td>
<td>One to One</td>
<td>10</td>
<td>11</td>
<td>Point-to-Point Channel Document Message</td>
<td>Manufacturer acknowledges and sends receipt of the order</td>
</tr>
<tr>
<td>8</td>
<td>One to One</td>
<td>11</td>
<td>12</td>
<td>Point-to-Point Channel Document Message</td>
<td>Manufacturer verifies whether ordered item is in stock in the inventory</td>
</tr>
<tr>
<td>9</td>
<td>One to Many</td>
<td>12</td>
<td>13, 18</td>
<td>Content-Based Router</td>
<td>Based on the stock levels for the item in the</td>
</tr>
<tr>
<td>S. No.</td>
<td>Interaction type</td>
<td>Initiator(s)</td>
<td>Responder(s)</td>
<td>Ideal patterns</td>
<td>Remarks</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>--------------</td>
<td>--------------</td>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td>10</td>
<td>One to Many</td>
<td>13</td>
<td>14, 22</td>
<td>Content-Based Router</td>
<td>If production order is not already placed then manufacturer places production order</td>
</tr>
<tr>
<td>11</td>
<td>One to One</td>
<td>18</td>
<td>19</td>
<td>Point-to-Point Channel</td>
<td>Retailer warehouse receives shipment from manufacturer and updates its stock levels</td>
</tr>
<tr>
<td>12</td>
<td>One to One</td>
<td>19</td>
<td>20</td>
<td>Point-to-Point Channel Document Message</td>
<td>Warehouse sends updated stock details and manufacture details to the retailer</td>
</tr>
<tr>
<td>13</td>
<td>One to One</td>
<td>14</td>
<td>15</td>
<td>Point-to-Point Channel Command Message</td>
<td>Production line to manufacture the ordered item is invoked</td>
</tr>
<tr>
<td>14</td>
<td>One to One</td>
<td>20</td>
<td>21</td>
<td>Point-to-Point Channel Document Message</td>
<td>Retailer sends payment details to manufacturer</td>
</tr>
<tr>
<td>15</td>
<td>One to One</td>
<td>15</td>
<td>16</td>
<td>Point-to-Point Channel Document Message</td>
<td>Upon completion of production, finished goods are moved to inventory</td>
</tr>
<tr>
<td>16</td>
<td>One to One</td>
<td>16</td>
<td>17</td>
<td>Transactional Client Document Message</td>
<td>Upon receipt of finished goods, inventory stock levels are updated.</td>
</tr>
</tbody>
</table>
**Type II and Type III errors listing**

**Type I (omission errors)** – missing ideal patterns or no pattern selected for interactions. All type I errors are major errors.

**Note:**
- If none of the selected pattern do not match with the below tables, then it would be considered as major type II error.
- Along with correct pattern selection, if there are additional pattern selected, then it would be considered as major error and evaluated on case by case basis to decide whether it is type II or type III error.

**Type II (Commission errors)** – incorrect patterns are selected. If a selected pattern is found in the below table in the selected pattern column against to the ideal pattern, then it should be considered as minor error.

<table>
<thead>
<tr>
<th>Ideal Pattern</th>
<th>Selected Pattern</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Expiration</td>
<td>Command Message</td>
<td>Message Expiration is used for sending information that time sensitive. Command Message is used for invoking an operation. Document Message is used for sending messages that not time sensitive, invoking an operation or event related Event Message is used for sending event related information or event trigger messages.</td>
</tr>
<tr>
<td></td>
<td>Document Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Event Message</td>
<td></td>
</tr>
<tr>
<td>Command Message</td>
<td>Event Message</td>
<td>Command Message is used for invoking an operation. Document Message is used for sending messages that not time sensitive, invoking an operation or event related Event Message is used for sending event related information or event trigger messages.</td>
</tr>
<tr>
<td>Document Message</td>
<td>Command Message</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Event Message</td>
<td></td>
</tr>
<tr>
<td>Event Message</td>
<td>Command Message</td>
<td></td>
</tr>
<tr>
<td>Datatype Channel</td>
<td>Message Filter</td>
<td>Datatype Channel is used for sending messages of particular type through a particular channel. Message Filter is router that eliminates undesired message from a channel based on preset criteria’s. Polling Consumer is used when receiver polls for message, processes it and then polls for another message. Receiver controls when each message in the channel gets consumed from the channel. Event-Driven Consumer is used to invoke operation upon a message</td>
</tr>
<tr>
<td></td>
<td>Polling Consumer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Event-Driven Consumer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selective Consumer</td>
<td></td>
</tr>
<tr>
<td>Message Filter</td>
<td>Datatype Channel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Polling Consumer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Event-Driven Consumer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selective Consumer</td>
<td></td>
</tr>
<tr>
<td>Ideal Pattern</td>
<td>Selected Pattern</td>
<td>Remarks</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Polling Consumer</td>
<td>Datatype Channel</td>
<td>delivery. Selective Consumer is used when receiver filters the messages delivered by a channel based on certain criteria’s, so that receiver consumes only desired messages.</td>
</tr>
<tr>
<td></td>
<td>Message Filter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Event-Driven Consumer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selective Consumer</td>
<td></td>
</tr>
<tr>
<td>Event-Driven Consumer</td>
<td>Datatype Channel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Message Filter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Polling Consumer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selective Consumer</td>
<td></td>
</tr>
<tr>
<td>Selective Consumer</td>
<td>Datatype Channel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Message Filter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Polling Consumer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Event-Driven Consumer</td>
<td></td>
</tr>
<tr>
<td>Publish-Subscribe Channel</td>
<td>Content-Based Router</td>
<td>Publish-Subscribe Channel delivers a copy of a message to set of receivers who have subscribed to receive particular messages.</td>
</tr>
<tr>
<td></td>
<td>Splitter</td>
<td></td>
</tr>
<tr>
<td>Content-Based Router</td>
<td>Publish-Subscribe Channel</td>
<td>Content-Based Router deliver messages to correct recipient based on message content.</td>
</tr>
<tr>
<td></td>
<td>Message Dispatcher</td>
<td>Message Dispatcher distributes a message to set of receivers and ensures that message is consumed only once by a receiver.</td>
</tr>
<tr>
<td></td>
<td>Recipient List</td>
<td>Recipient List delivers a copy of message to a set of predetermined receivers.</td>
</tr>
<tr>
<td></td>
<td>Splitter</td>
<td>Splitter publishes a composite message into individual messages.</td>
</tr>
<tr>
<td>Message Dispatcher</td>
<td>Content-Based Router</td>
<td>Durable Subscribe ensures that a message is consumed by a subscribe receiver by saving the message when receiver is disconnected.</td>
</tr>
<tr>
<td></td>
<td>Splitter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Durable Subscriber</td>
<td></td>
</tr>
<tr>
<td>Splitter</td>
<td>Publish-Subscribe Channel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Content-Based Router</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Message Dispatcher</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Recipient List</td>
<td></td>
</tr>
<tr>
<td>Durable Subscriber</td>
<td>Message Dispatcher</td>
<td></td>
</tr>
<tr>
<td>Aggregator</td>
<td>Content Enricher</td>
<td>Aggregator composes individual messages into a composite message.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content Enricher retrieves data from external sources and appends it to the message.</td>
</tr>
<tr>
<td>Content Enricher</td>
<td>Aggregator</td>
<td></td>
</tr>
</tbody>
</table>
Type III (Inconsistency errors) – inferior patterns were selected. If a selected pattern is found in the below table in the selected pattern column against to the ideal pattern, then it should be considered as minor error unless otherwise mentioned.

<table>
<thead>
<tr>
<th>Ideal Pattern</th>
<th>Selected Pattern</th>
<th>Error</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point to Point Channel</td>
<td>Datatype Channel</td>
<td>Major</td>
<td>Point-to-Point Channel is used for single one-way interaction.</td>
</tr>
<tr>
<td></td>
<td>Request Reply</td>
<td></td>
<td>Datatype Channel is used for multiple one-way channels each</td>
</tr>
<tr>
<td></td>
<td>Message Filter</td>
<td></td>
<td>dedicated for carrying a particular type of data.</td>
</tr>
<tr>
<td></td>
<td>Transactional Client</td>
<td></td>
<td>Request Reply is used for two way communication between</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>two partners. Partner 1 sends a request to Partner 2 and partner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 process it and sends reply to the Partner 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Message Filter is used as router to consume message that meets</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>certain criteria, rest of the messages are discarded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transactional Client is used when either receiver or sender is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sending or receiving transactional messages.</td>
</tr>
<tr>
<td>Datatype Channel</td>
<td>Point to Point Channel</td>
<td>Minor</td>
<td>Command Message is used for invoking an operation.</td>
</tr>
<tr>
<td>Request Reply Pattern</td>
<td>Point to Point Channel</td>
<td>Minor</td>
<td>Document Message is used for sending messages that not time</td>
</tr>
<tr>
<td>Message Filter</td>
<td>Point to Point Channel</td>
<td>Minor</td>
<td>sensitive, invoking an operation or event related.</td>
</tr>
<tr>
<td>Transactional Client</td>
<td>Point to Point Channel</td>
<td>Minor</td>
<td>Event Message is used for sending event related information or event</td>
</tr>
<tr>
<td>Command Message</td>
<td>Document Message</td>
<td>Minor</td>
<td>trigger messages.</td>
</tr>
<tr>
<td>Event Message</td>
<td>Document Message</td>
<td>Minor</td>
<td>Publish-Subscribe Channel delivers a copy of a message to set</td>
</tr>
<tr>
<td>Publish-Subscribe Channel</td>
<td>Message Dispatcher</td>
<td>Minor</td>
<td>of receivers who have subscribed to receive particular messages.</td>
</tr>
<tr>
<td></td>
<td>Recipient List</td>
<td></td>
<td>Message Dispatcher distributes a message to set of receivers and</td>
</tr>
<tr>
<td>Message Dispatcher</td>
<td>Publish-Subscribe Channel</td>
<td>Minor</td>
<td>ensures that message is consumed only once by a receiver.</td>
</tr>
<tr>
<td>Recipient List</td>
<td>Publish-Subscribe Channel</td>
<td>Minor</td>
<td>Recipient List delivers a copy of message to a set of predetermined</td>
</tr>
<tr>
<td></td>
<td>Message Dispatcher</td>
<td></td>
<td>receivers.</td>
</tr>
</tbody>
</table>
KARTHIKEYAN UMAPATHY
www.karthikeyan.umapathy.com

Education

Ph.D. Information Sciences and Technology (May 2008), Pennsylvania State University
M.S Industrial Engineering (May 2003), Pennsylvania State University
B.E. Mechanical Engineering (May 1998), PSG College of Technology, Coimbatore, India

Journals


Conferences


Reports