Services Computing

Liang-Jie Zhang
Jia Zhang
Hong Cai

TSINGHUA UNIVERSITY PRESS
Springer

Liang-Jie Zhang
Jia Zhang
Hong Cai

Services Computing

Liang-Jie Zhang
Jia Zhang
Hong Cai

Services Computing

With 218 figures

TSINGHUA UNIVERSITY PRESS

Springer

Liang-Jie Zhang
Ph.D., Research Staff Member
IBM T.J. Watson Research Center
19 Skyline Drive, Hawthorne,
NY 10532, USA
Email: zhanglj@us.ibm.com
http://www.research.ibm.com/people/z/zhanglj8

Jia Zhang
Ph.D., Assistant Professor
Department of Computer Science
Northern Illinois University
DeKalb, IL 60115, USA
Email: jiazhang@cs.niu.edu
http://www.cs.niu.edu/~jiazhang

Hong Cai
Ph.D., Research Staff Member
Services Research
IBM China Research Lab
Zhongguancun Software Park
Haidian District
Beijing 100094, P.R. China
Email: caihong@cn.ibm.com

ISBN 978-7-302-15075-6 **Tsinghua University Press, Beijing**
ISBN 978-3-540-38281-2 **Springer Berlin Heidelberg New York**

Library of Congress Control Number: 2007930066

Co-published by Tsinghua University Press, Beijing and Springer-Verlag GmbH Berlin Heidelberg

Springer is a part of Springer Science+Business Media
springer.com

Cover design: Frido Steinen-Broo, EStudio Calamar, Spain
Printed on acid-free paper

Preface

Services Computing has become a cross-discipline subject that covers the science and technology of bridging the gap between Business Services and IT Services. Its most recent enabling technology is Web services centered on Service-Oriented Architecture (SOA). This book depicts an overall picture of the state-of-the-art of the field. A comprehensive set of innovative research results and solution methods is described and discussed, including business componentization, services modeling, services creation, services realization, services annotation, services deployment, services discovery, services composition, services delivery, service-to-service collaboration, services monitoring, services optimization, services management, business consulting methodology and utilities, business process modeling, transformation, integration, and management.

What Is the Uniqueness of This Book?

This book introduces innovative ideas and solutions based on existing key techniques and industry standards in the field of Services Computing. In particular, this book illustrates Services Computing as an emerging interdisciplinary science and engineering subject bridging the gap between business/application services and IT services. It presents a lifecycle view of modern services industry, discusses up-to-date innovative research directions and industry solutions in the domain of Services Computing. It explains how to effectively and efficiently establish, operate, and manage business and application services using Services Computing, and it also guides research directions of Services Computing.

There have been numerous publications in the market regarding Web services and Service-Oriented Architecture from specifications and standards perspectives. To our knowledge, however, this book is the first that provides a systematic view of SOA solutions and SOA services to enable the lifecycle of modernized services businesses and applications. In our view, Services Computing is not merely a technical direction;

instead, it is an interdisciplinary area aiming to bridge the gap between IT and business. Therefore, it is not only necessary, but also critical to consider Services Computing from strategic point of view. Moreover, Service-Oriented Computing is just a pure technology fraction of Services Computing that also includes services consulting methodologies, services design and service delivery, as well as services maintenance and management.

As for implementation, there have emerged a number of industry standards and specifications for Web services, such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), Universal Description, Discovery, and Integration (UDDI), Business Process Execution Language for Web Services (BPEL4WS). However, from our point of view, these existing specifications are just examples of infrastructure enabling technologies for Services Computing environment. With the development of Services Computing, these specifications and technologies will continuously be evolving into their next generation or be replaced by new technologies and standards.

Throughout this book, instead of repeating the existing specifications, we concentrate on introducing innovative frameworks and methods on how to leverage related technologies to address real business challenges. The existing technologies are used as examples to study the state-of-the-art of the field and can be used as starting points for further innovations. Along with the newly introduced ideas in this book, the present enabling technologies provide a comprehensive framework that can be used to construct domain-specific SOA solutions. It should be noted that the existing technologies may have to be adjusted, extended, and customized in accordance with particular execution contexts and business requirements.

Finally, this is a foresighted book intended to spur researchers, practitioners, and students into further explorations and investigations in the field of Services Computing. As SOA and services engineering become mainstream, there are numerous efforts underway in both academia and industry, all of which deliver concepts and technologies in the same or similar fashion. This book aims to guide readers to grasp the foundations and state-of-the-art developments in the field of Services Computing.

Who Should Read This Book?

Researchers and students

The audience first includes researchers, graduate students, and senior undergraduates who seek a systemic introduction to the key technologies and research innovations in the field of Services Computing. This book can be used as an introductory textbook, advanced undergraduate textbook, graduate textbook, continuing education textbook (e.g., for executive MBA), or supplemental reading materials in classrooms.

In addition, this book can be used as a reference book on advanced technologies for a set of existing courses such as Modern Software Engineering, Web Engineering,

Web Technologies, Advanced Software Engineering Methodologies, Advanced Software Architecture, and so on. Targeting departments include Department of Computer Science, Department of Industrial Engineering, Department of Business Management, Department of Automation, and Department of Management of Information System.

This book is organized in a way that is suitable for students to learn the Services Computing concepts and technologies step by step. It is written in a way that it can be used as a classroom textbook, as well as a self-study reference book.

Engineers and managers

As Services Computing is being widely accepted by the business world, practitioners who are interested in building value-added services or solutions based on SOA will become suitable audiences. Companies that either develop software using SOA or intend to introduce SOA and Web services in business could use this book as a reference book for their software engineers, IT managers, business managers, salesmen, and IT and business executives.

Outline of This Book

Part 1: Foundations of Services Computing

This part introduces core techniques of Web services modeling, registry, and discovery. SOA paradigm is discussed, along with SOA solution architecture based on industry best practices. Current SOA and Web services standard stack is also presented. Advanced techniques are introduced including multi-dimensional services modeling, dynamic services invocation, federated services discovery, services relationship modeling, and solution-level Quality of Service (QoS) in SOA.

Part 2: Realization of Services Computing

This part introduces services realization technologies from four perspectives: requirements-driven services composition, services value chain collaboration, business process management and integration, as well as business grid.

Part 3: Services Delivery and Services Engineering

This part introduces technologies and methodologies for services delivery and engineering from four perspectives: project-based business performance management, service-oriented business consulting methodology, end-to-end services delivery

platform and methodology, as well as software as services and services as software.

Acknowledgements

All the authors would like to acknowledge the invaluable cooperation of many collaborators who have been involved in the research projects in the past years. Those projects served as a basis for some research findings that underlie several of the ideas discussed in this book.

Liang-Jie (LJ) Zhang would like to send his sincere appreciation to Fausto Bernardini for his sturdy support and insightful directions. Special thanks also go to his management team: Robert Morris, Alain Azagury, Manoj Kumar, and David Cohn.

LJ would like to send special thanks to his previous collaborators in the past years: Henry Chang, Tian Chao, Jen-Yao Chung, Bing Li, Haifei Li, Yu Long, John Y. Sayah, Jing Min Xu, Shun Xiang Yang, Qun Zhou, and Ying Nan Zuo.

LJ would also like to appreciate his IBM SOA Solution Stack core team members for their insightful discussions: Ali Arsanjani, Abdul Allam, Kishore Channabasavaiah, and Michael Ellis. He would like to acknowledge the technical advice from Kerrie Holley, Ray Harishankar, George Galambos, and Shankar Kalyana during the past SOA projects, as well as valuable technical discussions with Sridhar Iyengar and Ed Kahan.

Finally, LJ would like to send his special thanks to Cesar A. Gonzales, Maria Azua, and Daniel Sturman for their invaluable encouragement and advice over the years.

Jia Zhang would like to send special thanks to Dr. Raimund K. Ege for his invaluable support.

The authors would also like to thank Annie Wang for helping proofreading the book.

Of course, our most profound thanks go to our families for their continuous love and encouragement.

To my wife, Qun, and my two daughters, Lan and Anna, with all my love.

Liang-Jie Zhang

To my dearest parents for their constant love and inspiration in my life.

Jia Zhang

I would like to express my sincere thanks to my dear wife Yi for her love and patience over the many evenings and weekends when I was completing this book and was unable to spend time with her and little daughter LuYi.

Hong Cai

Contents

Part 1 Foundations of Services Computing

1 The Principle of Services and Services Computing 3
1.1 Introduction of Services 3
1.1.1 Definition of Services 3
1.1.2 Definition of Services System 4
1.2 Perspectives of Services Systems 6
1.2.1 Model 6
1.2.2 Technology 7
1.2.3 Architecture 7
1.2.4 Optimization 7
1.3 Services Lifecycle 7
1.3.1 Phase 1: Consulting and Strategic Planning 8
1.3.2 Phase 2: Services Engagement 8
1.3.3 Phase 3: Services Delivery 8
1.3.4 Phase 4: Services Operation 9
1.3.5 Phase 5: Services Billing 9
1.3.6 Phase 6: Services Management 9
1.4 Key Factors in a Services Lifecycle 9
1.4.1 Data/Information 9
1.4.2 Processes 10
1.4.3 People 10
1.4.4 Resources 10
1.4.5 Finance Factors 11
1.4.6 Knowledge and Skills 11
1.4.7 Innovation and Technology 11
1.5 Comparisons Between Services and Manufacturing Models 11
1.6 Business Services Lifecycle in Telecommunication Industry 13
1.7 Relationship Between IT and Non-IT Services in a Services Ecosystem 14
1.7.1 Overview of a Services Ecosystem 14
1.7.2 Comparisons Between IT and Non-IT Services 15
1.8 Emergence of Services Computing Technology 17
1.9 Summary 18
References 19

2 e-Business Evolution 20
2.1 Phases of e-Business Adoption 20
2.2 Top Trends of e-Business 22
2.3 IT Innovations to Flatten the World 22
2.3.1 Interactive Multimedia Services 22
2.3.2 Office Online Services 25
2.3.3 Globalization of Businesses 26
2.4 "Open" Trends for Technologies and Services Ecosystems 27
2.4.1 Open Standards 28
2.4.2 Open Sources 28
2.4.3 Open Architecture: Service-Oriented Architecture (SOA) 29
2.5 Debuts of New Service-Oriented Business Models 31
2.5.1 Services Modernization 31
2.5.2 Software as Services 32
2.5.3 Services as Software 33
2.6 New Discipline: Services Computing 34
2.7 Summary 35
References 36

3 Web Services Modeling 37
3.1 Basic Concept of Web Services 37
3.2 Modeling of a Web Service 38
3.2.1 Basic Concepts of WSDL Modeling 38
3.2.2 Web Services Communication Protocol: SOAP 41
3.2.3 Binding of WSDL to SOAP 44
3.2.4 Publishing a Web Service in Registry 45
3.2.5 Stateful Web Services Modeling 46
3.2.6 Web Services Interoperability 50
3.3 Modeling a Composite Web Service 50
3.3.1 Basic Concepts of BPEL 51
3.3.2 BPEL Basic Structure: via an Example 51
3.3.3 BPEL Key Elements 57
3.4 Three-Dimensional Web Services Modeling 59
3.5 Discussions on Web Services Modeling 60
3.6 Summary 62
References 62

4 Web Services Publishing and Discovery 64
4.1 Web Services Publishing 64
4.1.1 Public/Private UDDI Publishing 64
4.1.2 WSIL Publishing 65
4.1.3 UDDI Publishing *vs*. WSIL Publishing 67

4.2 Simple Web Services Discovery 67
4.2.1 Simple UDDI Search 68
4.2.2 Simple WSIL Search 70
4.2.3 Issues of the Simple UDDI/WSIL Search 70
4.3 UDDI Search Markup Language 71
4.3.1 USML Schema 72
4.3.2 Composite Search Options 74
4.3.3 Aggregation Operators 75
4.4 USML-Based Advanced UDDI Search Engine (AUSE) 76
4.4.1 AUSE Structure 77
4.4.2 AUSE-Based Search Process 78
4.5 WSIL-Oriented Dynamic Services Discovery Framework (DSDF) 80
4.5.1 Architecture of DSDF 80
4.5.2 DSDF Implementation 83
4.6 Federated Web Services Discovery Framework 83
4.6.1 Basic Ideas 83
4.6.2 Search Language 85
4.6.3 Comparison with Generic Web Search Engine 86
4.7 Discussions on Web Services Publishing and Discovery 87
4.8 Summary 88
References 88

5 Service-Oriented Architecture 89
5.1 Concept of Service-Oriented Architecture (SOA) 89
5.1.1 Triangular SOA Operational Model 89
5.1.2 Web Services-Based SOA 90
5.2 Services Invocation 91
5.2.1 Simple Services Invocation 91
5.2.2 Introduction to MetaWSDL 93
5.2.3 MetaWSDL Publishing 97
5.2.4 MetaWSDL-Based Advanced Services Invocation Framework 98
5.3 SOA: Bridging Business and IT Architecture 99
5.4 SOA Solution Lifecycle 101
5.4.1 Modeling 102
5.4.2 Development 103
5.4.3 Deployment 103
5.4.4 Publishing 103
5.4.5 Discovery 104
5.4.6 Invocation 104
5.4.7 Composition 104
5.4.8 Collaboration 105
5.4.9 Monitoring and Management 105

5.5 Enterprise Service Bus (ESB) 106
5.6 SOA Reference Architecture (SOA-RA) 107
5.7 Discussions on SOA 111
5.8 Summary 111
References 112

6 Services Relationship Modeling 114
6.1 Introduction to Services Relationship Modeling 114
6.1.1 UDDI Specifications on Simple Relationships 115
6.1.2 Other Relationship Specification Languages 115
6.2 Web Services Relationship Language (WSRL) 117
6.2.1 Structure of a WSRL Document 117
6.2.2 WSRL Discussions 117
6.3 Layered Services Relationship Modeling 119
6.4 SOA-Based Relationship Modeling Language (SOA-RML) 120
6.4.1 Business Services Relationships at Business Entity Level 120
6.4.2 Business Services Relationships at Business Service Level 125
6.4.3 Layered Relationships Summary 127
6.5 SOA-RML-Enriched Services Registry 130
6.5.1 SOA-RML Schema 131
6.6 Discussions on SOA-Based Relationship Modeling 132
6.7 Summary 132
References 133

7 SOA and Web Services Standards 134
7.1 Introduction 134
7.2 Web Services Standard Stack 134
7.2.1 Transport 135
7.2.2 Messaging 137
7.2.3 Description/Publishing/Discovery 138
7.2.4 Quality of Service (QoS) 141
7.2.5 Service Composition 144
7.3 Industry-Specific Service-Oriented Standards 145
7.3.1 Electronics Industry 145
7.3.2 Insurance Industry 146
7.3.3 Telecommunication Industry 147
7.4 Generic SOA Solution Standards are Evolving 148
7.5 Discussions on SOA and Web Services Standards 149
7.6 Summary 149
References 150

8 Solution-Level Quality of Service in SOA 152
8.1 State-of-the-art of QoS on Web Services 152

8.2 SOA-QoS 153
8.2.1 Context-Aware QoS Model 153
8.2.2 Representation of QoS Model 154
8.2.3 QoS Data Management 157
8.2.4 Business Relationship Model 157
8.3 QoS Framework in an SOA Solution 158
8.3.1 QoS Framework Descriptions 158
8.3.2 Relationships Between Constructs in QoS Framework 161
8.4 Data Architecture Framework 162
8.4.1 Data Architecture Framework Descriptions 162
8.4.2 Relationships Between Constructs in Data Architecture 164
8.5 Modeling the Key Elements in QoS Management 165
8.5.1 Modeling of Resources 165
8.5.2 Modeling the QoS Assurance Process 167
8.6 Discussions on QoS in SOA 169
8.7 Summary 169
References 169

Part 2 Realization of Services Computing

9 Requirements Driven Services Composition 173
9.1 Introduction 173
9.2 Business Requirements Modeling 174
9.2.1 Target Components and Environment 175
9.2.2 Asset Lifecycle Management 177
9.2.3 Project Management 177
9.2.4 Finance Management 178
9.2.5 Representation of Business Requirements Modeling 178
9.3 Requirements Driven Services Discovery 180
9.4 Optimization for Business Services Composition 183
9.4.1 Formalization of Business Services Composition 183
9.4.2 Optimization Algorithms for Business Services Composition 188
9.5 Service Integration Framework 190
9.5.1 Services Integration Procedure 191
9.6 Discussions on Services Composition 192
9.7 Summary 193
References 194

10 Services Value Chain Collaboration 195
10.1 Value Chain Collaboration 195
10.1.1 Example of Business Collaboration 195

10.1.2 Inter- and Intra-Enterprise Collaboration 197
10.1.3 Web Services Based Value Chain Collaboration 199
10.2 Extended Business Collaboration (eBC) Model 199
10.2.1 Introduction to Business Resources 199
10.2.2 Annotated Business HyperChain Technique 201
10.2.3 eBC to WS-Collab 201
10.3 Web Services Collaboration (WS-Collab) Resources 202
10.3.1 WS-Collab Resources 202
10.3.2 WS-Collab Resource Specifications 204
10.3.3 WS-Collab Ontology on Relationships Between Resources 205
10.4 Web Services Collaboration Message Primitives 211
10.4.1 WS-Collab Primitive 211
10.4.2 WS-Collab Message Structure 213
10.5 Web Services Collaboration Construct 215
10.6 Web Services Collaborative Exchange Protocol 217
10.7 WS-Collaboration Realization 219
10.7.1 Annotation Data Generation Process 219
10.7.2 HyperChain Manager 220
10.8 Relationships with Industry Standards 221
10.9 Discussions on Service-Based Business Collaboration 222
10.10 Summary 222
References 222

11 Business Process Management and Integration 224
11.1 Business Process Modeling 224
11.2 SOA-Based Business Process Management 225
11.2.1 Top-down Business Process Management 226
11.2.2 Bottom-up Business Process Management 227
11.3 Bridging Gap Between Business Modeling and IT Implementation 228
11.3.1 Business Process Modeling from Business Analysts 228
11.3.2 Business Process Re-engineering in SOA 229
11.3.3 Generic Guidance for Re-engineering Business Process Modeling in SOA 230
11.3.4 Methodology for Re-engineering Business Process Models in SOA 230
11.4 Flexible Business Process Integration in SOA 234
11.4.1 Integration Ontology 234
11.4.2 Integration Manager 238
11.4.3 Lifecycle of an Integration Activity 239
11.4.4 Business Process Monitoring 240
11.5 Discussions on Business Process Management and Integration 241

11.6 Summary 242
References 242

12 Business Grid 243
12.1 Grid Computing 243
12.2 Open Grid Services Architecture (OGSA) 244
12.2.1 Distributed Resource Sharing Using OGSA 245
12.3 Business Grid 248
12.3.1 Enhancing OGSA with Advanced Web Services Technologies 248
12.3.2 Concept of Business Grid 249
12.3.3 Business Grid Solution Framework 250
12.4 Logical Grid Infrastructure 251
12.4.1 Packaged Application Grid 251
12.4.2 Business Grid Middleware 252
12.4.3 Business Process Grid 253
12.5 Business Grid Service Development and Invocation 254
12.6 Discussions on Business Grid 255
12.7 Summary 255
References 255

Part 3 Service Delivery and Services Engineering

13 Enterprise Modeling 259
13.1 Introduction 259
13.1.1 Dynamics of Services Ecosystem 259
13.1.2 Requirements from Decision Makers 260
13.2 Methodologies for Enterprise Modeling 261
13.2.1 Balanced Scorecard and Strategy Map 261
13.2 2 Component Business Modeling Circle 264
13.2.3 Enterprise Architecture 269
13.2.4 Relationships Between Enterprise Models and Business Process Transformation Model 272
13.3 Discussions on Enterprise Modeling 273
13.4 Summary 273
References 274

14 Project Based Enterprise Performance Management 275
14.1 Changes of Enterprise Operational Views 275
14.2 Overview of Project Management 277
14.3 Enterprise Performance Management (EPM) 279
14.3.1 Concept of EPM 279

14.3.2 EPM Framework 279
14.3.3 From Project Management to Enterprise Portfolio Management 280
14.4 Service-Oriented Enterprise Project Management 281
14.4.1 EPM toward SOA 281
14.4.2 WS-EPM Framework 282
14.4.3 WS-EPM Operations 284
14.4.4 Formalization of WS-EPM Methodology 285
14.5 WS-EPM Common Services 286
14.5.1 WS-EPM Resource Management Facility 286
14.5.2 WS-EPM Utilities 290
14.6 WS-EPM Workspace 292
14.7 Discussions on SOA-Based EPM 294
14.8 Discussions on Enterprise Portfolio Management 294
14.9 Summary 295
References 295

15 Service-Oriented Business Consulting Methodology 296
15.1 Vision of Services System 296
15.2 The Traditional Business Consulting Methods 298
15.2.1 Traditional Consulting Method for Strategic Change 298
15.2.2 Traditional Consulting Method for IT Strategic Plan 298
15.2.3 Shortcomings of Traditional Methods 299
15.3 Modeling of Services Ecosystem 299
15.4 Service-Oriented Business Consulting Method 301
15.4.1 Gap Analysis over SOA 301
15.4.2 Identification of Transformation Initiatives 303
15.4.3 Value Chain Analysis 304
15.4.4 Business Case Analysis 305
15.4.5 Portfolio Analysis and Transition Planning 306
15.4.6 Service-Oriented Project Management and Collaboration 307
15.4.7 IT Service Management 307
15.5 Discussion on SO-BCM 308
15.6 Summary 308
Reference 308

16 End-to-End Services Delivery Platform and Methodology 310
16.1 Introduction to Services Delivery 310
16.2 Changes of Services Delivery Mechanisms 310
16.3 An SOA-Based Services Delivery Platform 312
16.3.1 Layered View of the Services Delivery Platform 312
16.3.2 Collaboration View of the Services Delivery Platform 314
16.3.3 Key Services Needed in the Services Delivery Platform 317

16.4 The End-to-End Services Delivery Methodology 322
16.4.1 Services Delivery Readiness Phase 323
16.4.2 Services Delivery Creation Phase 324
16.4.3 Services Delivery Operation Phase 326
16.5 Discussions on the Services Delivery Methodology and Platform 327
16.6 Summary 328
References 328

17 Software as Services and Services as Software 330
17.1 Software as Services 330
17.1.1 Next Generation of Internet: Web 2.0 331
17.1.2 A Case Study of Web 2.0 Service Model—Service Mash-up 334
17.1.3 New Business Models Through Software as Services 335
17.1.4 Tips for Software as Services Model 336
17.2 Services as Software 336
17.3 Successful Business Cases 337
17.3.1 Healthcare Under Transformation 338
17.3.2 Innovative Store 340
17.3.3 Personalized Insurance Premiums 340
17.3.4 Business Performance Transformation Services 342
17.4 Summary 343
References 343

Index 345

Part 1 Foundations of Services Computing

1 The Principle of Services and Services Computing

1.1 Introduction of Services

1.1.1 Definition of Services

The term "service" has existed for thousands of years along human history. When a person or a group performs some work to benefit another, it becomes a service. Many versions of definitions exist for the term "service". For example, James Fitzimmons[1] defines a service as follows:

> *"A service is a time-perishable, intangible experience performed for a customer acting in the role of co-producer."*

Christian Gronroos defines a service from the perspective of management and marketing as follows[2]:

> *"A service is an activity or series of activities of more or less intangible nature that normally, but not necessarily, take place in interactions between customer and service employees and/or physical resources or goods and/or systems of the service provider, which are provided as solutions to customer problems."*

Although these definitions look different, they all indicate a fact that each service involves two inevitable sides: service provider and service consumer. A service provider offers the service, and a service consumer utilizes the service. The interaction between a service consumer and a service provider may happen in real-time or off-line. Focusing on IT-enabled business services, this book defines the term "*services"* as follows:

> *Services represent a type of relationships-based interactions (activities) between at least one service provider and one service consumer to achieve a certain business goal or solution objective.*

A service provider commits to complete the tasks and provide values to a service consumer during the service's lifecycle. Both sides share a common goal of keeping a healthy, long-term trust with efficient and valuable services.

An IT-enabled business service is typically characterized by two features: its service operation model and its service charge model. A service operation model defines how the service is to be delivered; a service charge model specifies how

SOMA-ME: A platform for the model-driven design of SOA solutions

L.-J. Zhang
N. Zhou
Y.-M. Chee
A. Jalaldeen
K. Ponnalagu
R. R. Sindhgatta
A. Arsanjani
F. Bernardini

The service-oriented modeling and architecture modeling environment (SOMA-ME) is first a framework for the model-driven design of service-oriented architecture (SOA) solutions using the service-oriented modeling and architecture (SOMA) method. In SOMA-ME, Unified Modeling Language (UML™) profiles extend the UML 2.0 metamodel to domain-specific concepts. SOMA-ME is also a tool that extends the IBM Rational® Software Architect product to provide a development environment and automation features for designing SOA solutions in a systematic and model-driven fashion. Extensibility, traceability, variation-oriented design, and automatic generation of technical documentation and code artifacts are shown to be some of the properties of the SOMA-ME tool.

INTRODUCTION

Service-oriented architecture (SOA) is an information technology (IT) architectural approach that supports the creation of business processes from functional units defined as services.[1,2] It has become a major focus in the emerging services computing discipline, which explores the ways in which IT can be used to develop and manage business processes efficiently.[3] Helping customers implement SOA solutions, however, involves some major challenges. How do we develop an SOA solution in a way that ensures the reusability of the software artifacts developed? How do we ensure that the solution we develop is extensible? How do we develop software tools that validate an SOA solution so that we reduce the cost of maintaining it over its lifetime?

In this paper, the service-oriented modeling and architecture modeling environment (SOMA-ME) is presented as a platform for addressing the above challenges. SOMA-ME is first a framework for the model-driven design of service-oriented architecture (SOA) solutions using service-oriented modeling and architecture (SOMA) or similar methods.[4,5] It is also a tool that extends the IBM Rational* Software Architect (RSA) product to provide a development environment and automation features for designing SOA solutions in a systematic and model-driven fashion.[6]

The SOMA-ME tool offers context-aware guidance during the SOMA-based design process, reusable SOMA design patterns, a customizable algorithm for selecting the set of services to be exposed (to be used for composing other services), support for the automated generation of technical documentation and code artifacts, and a graphical user interface (GUI) customization engine for adapting the tool to new business requirements. After the SOA solution model is strengthened by populating its variables, a validation module can be triggered to check the captured models for correctness and completeness, using validation rules based on best practices in the field.

A considerable amount of work has been performed in federated discovery of Web services and dynamic Web services composition, and related products and standards are being developed. IBM Web Services Outsourcing Manager (WSOM) enables the dynamic composition of service-oriented business process flow based on customer requirements.[7] An XML-based (Extensible Markup Language-based) business process outsourcing language was proposed for capturing customer requirements. The requirements are analyzed and a search script for automatically finding services is generated; services are then composed for performing the required task.[8] WSOM makes use of Business Explorer for Web Services, a federated Web services discovery engine.[9,10] A formal approach to Web services identification, based on analyzing object models, was proposed in Reference 11.

In this paper, SOMA-ME has extended WSOM by using model-driven architecture and best practices of designing SOA solutions. This object model-based analysis for Web services discovery could be leveraged in the proposed SOMA-ME to support its services identification phase.[11]

A new Web Services Description Language (WSDL) metamodel that is QoS-enabled (quality of service-enabled) was introduced in Reference 12. A model-driven process for Web service development that includes the transformation of WSDL to Unified Modeling Language (UML**) was proposed in Reference 13. Both of these papers focus on the model-driven design for WSDL-based development of Web services. A Service Component Architecture-based (SCA-based) UML profile was used to model services and their extra-functional properties.[14] A novel, model-driven approach, based on UML 2.0, was developed, which takes existing Web service interfaces as input and generates an executable Web service composition.[15] As we know, the WSDL file is just one type of solution artifact in an SOA solution. None of these cited works has been extended to integrate business goals, business processes, WSDL interfaces, data types, and architectural building blocks, which are the core elements of the model-driven approach to developing SOA solutions presented in this paper.

A visual approach, based on the use of software models and graph transformations, was proposed to identify the services matching a set of given requirements.[16] An architecture-driven service discovery framework was proposed that aims to satisfy constraints specified in the design phase.[17] It includes a query extractor for handling structural UML design models and a query execution engine that handles queries based on graph-matching techniques. Those two research results provide service-matching techniques that can be applied to the identification of solution artifacts based on service properties and UML-based model representations in the SOMA-ME framework presented in this paper.

The paper is organized as follows. In the next section we review the current practice for developing SOA solutions. In the section that follows we discuss UML profiles and the key components of SOMA-ME. In the next section, the functions of the tool are presented and illustrated with examples. These include: context-aware guidance through the SOMA steps, reusable SOMA design patterns, a customizable algorithm for selecting the services to be exposed (to be used for generating other services by composition), modeling of the SOA solution architecture, a pattern recognition-based facility for generating technical documentation and code artifacts, and achieving traceability through domain-specific impact analysis. The paper concludes with a discussion of future directions and conclusions.

SOA SOLUTION DESIGN PROCESS: CURRENT PRACTICE

Figure 1 illustrates the current practice for designing and managing SOA solutions.[18] This generic SOA solution process, which covers all phases of the SOA solution life cycle, can be customized for different application domains.

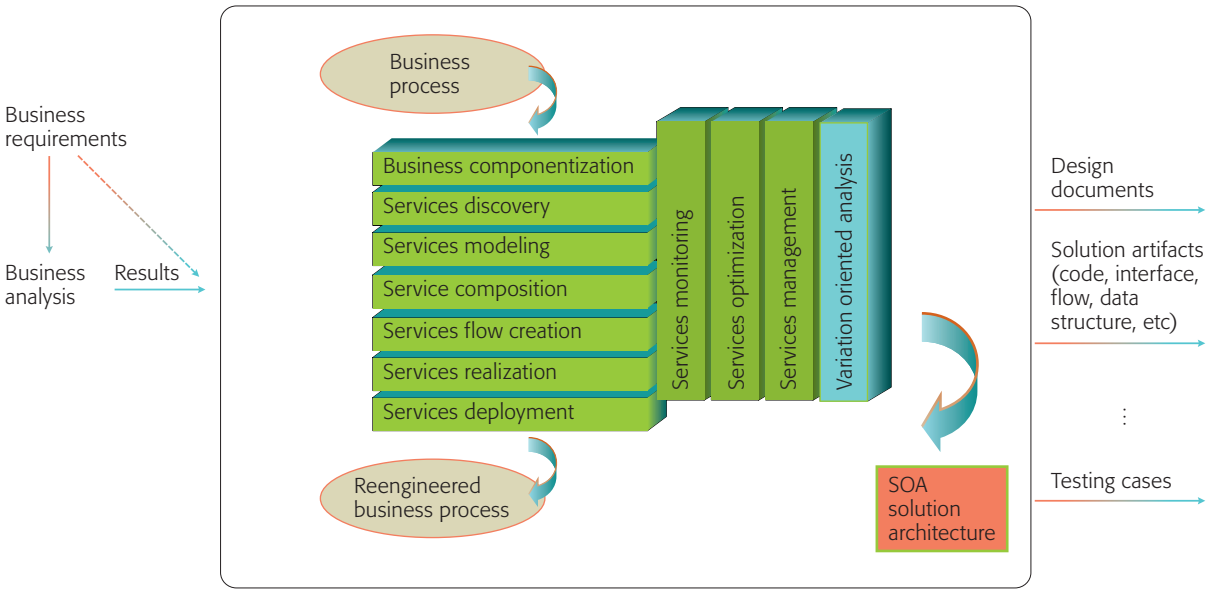


Figure 1
Current practice for designing SOA solutions

FPO

The horizontally placed steps in Figure 1 (labeled "SOA solution design") define a set of solution artifacts and configure these into a solution that satisfies the business requirements. Each of these steps may involve a number of iterations. The input to these steps includes business processes, business requirements, and the results of business analysis performed by business analysts or SOA business architects. *Business componentization* derives the major business domains of an enterprise (e.g., human resources, supply chain, and research and development), each of which is supported by a set of business processes and a set of requirements and objectives. Business functions within each domain are also identified in this phase. *Service discovery* involves the search for and the identification of candidate services that can be used to implement business functions and thus realize the business objectives.[10,11] *Service modeling* involves building models of candidate services in order to analyze the interactions between them. *Service composition* involves the assembly of candidate services according to the composition rules identified in the service modeling step, resulting in the creation of composite services.[8] *Services flow creation* involves the creation of high-level business process flows from composite services, individual services, and control flows (i.e., the order in which services are invoked).[8] *Services realization* involves the implementation of services based on a selected technology, such as Java** classes, Enterprise JavaBeans,** or Microsoft .NET** components. *Service deployment* involves the deployment of services onto Web application server platforms.

The vertically oriented steps (e.g., services monitoring) include activities that relate to quality of service (QoS) goals and are carried out throughout the entire SOA solution life cycle. *Service monitoring* involves monitoring the performance of services to ensure that it meets the QoS requirements. *Service optimization* involves optimizing performance of services, service selection rules, or service usage patterns. *Service management* involves managing and documenting the behavior of services to support reusability, flexibility, and traceability (traceability of requirements means the ability at any time to determine where the requirements are derived from, how they are satisfied, how they are tested, and what will be the impact of changing them). *Variation-oriented analysis* involves identifying, providing for, and incorporating variations on the SOA solution during its lifetime.[18]

Following the completion of the SOA solution design process, the business processes are reengineered into service-oriented business processes. In addition, the SOA solution design process also produces

an SOA solution architecture (Figure 1). A set of documents describing various aspects of the SOA solution architecture is manually generated. A selected set of solution artifacts (for example, code skeletons, data structures, control flows, interfaces, etc.) used in the SOA solution architecture could be packaged for further development so as to conform to industry norms such as Service Component Architecture (SCA).[19] Test cases for unit testing, functional testing, and system testing are manually generated.

UML PROFILES AND KEY COMPONENTS OF SOMA-ME

SOMA-ME is based on the model-driven approach to developing end-to-end SOA solutions. It uses UML[20] as the language for modeling software systems. The UML metamodel, a description of UML in UML, describes the objects, attributes, and relationships necessary to represent UML concepts within a software application. UML *profiles* provide a means to extend the metamodel with domain-specific concepts. These concepts capture additional information about services, such as origins, associated goals, key performance indicators (KPIs), nonfunctional requirements, and interdependencies among services. In SOMA-ME, the profiles are contained in two packages: the *SOA method-profile package* and the *SOA architecture-profile package*.

The SOA method-profile package includes an *SOA business processes profile*, an *SOA services profile*, and an *SOA service components profile*. The SOA business processes profile contains stereotypes (a UML extension mechanism) for concepts used in service identification techniques (such as domain decomposition, goal-service modeling, and existing asset analysis). The SOA services profile contains stereotypes that address the service identification and service specification steps. Finally, the service components profile captures information related to service realization. These UML profiles were first presented in Reference 18; we present here an enhanced version. The SOA architecture-profile package, which defines UML profiles to capture the entities and associations in the SOA solution architecture, is discussed elsewhere.[21]

Figure 2 illustrates a metamodel that describes the attributes and relationships among SOA solution artifacts in SOMA-ME. It is noted that the SOMA-ME profiles extend the UML metamodel of the UML 2.0 profile for software services[22] with those constructs that are defined by a generic SOA solution design method such as SOMA. As a subset of the SOA method-profile package, the IBM Rational UML 2.0 profile for software services only covers Web services-level services specification, operations, and messages. The UML profiles for representing SOA solution artifacts in SOMA-ME are defined as follows.

`<<Stereotype>> Domain` corresponds to a logical grouping of business capabilities. During domain decomposition, Domain model elements are created to represent each domain defined within the SOA solution.

`<<Stereotype>> FunctionalArea` represents a grouping of business functionality consistent with the business analysts' view of their business domain. During functional area analysis, functional areas are created under each domain to decompose the domain into its major functional responsibilities.

`<<Stereotype>> Function` represents a business function that is provided by a functional area. The `<<Function>>` stereotype is used to associate candidate services identified as functions during functional area analysis with their corresponding function.

`<<Stereotype>> Process` represents a business process or part of a business process (a subprocess) that is decomposed in the service identification step of the SOA solution design. `<<Process>>` is used to model the set of processes and subprocesses identified during the process decomposition activity of domain decomposition.

`<<Stereotype>> Metric` represents an indicator variable used to determine the attainment level of a business goal.

`<<Stereotype>> KPI` represents a key performance indicator, a quantifiable objective used to measure progress against business goals.

`<<Stereotype>> ExistingAsset` models an existing application or subsystem that can be used in an SOA solution. It identifies the functions of an existing asset and the metrics associated with these functions. This stereotype is meant to be applied in addition to any other applicable stereotypes; it tags a model element as representing an existing asset

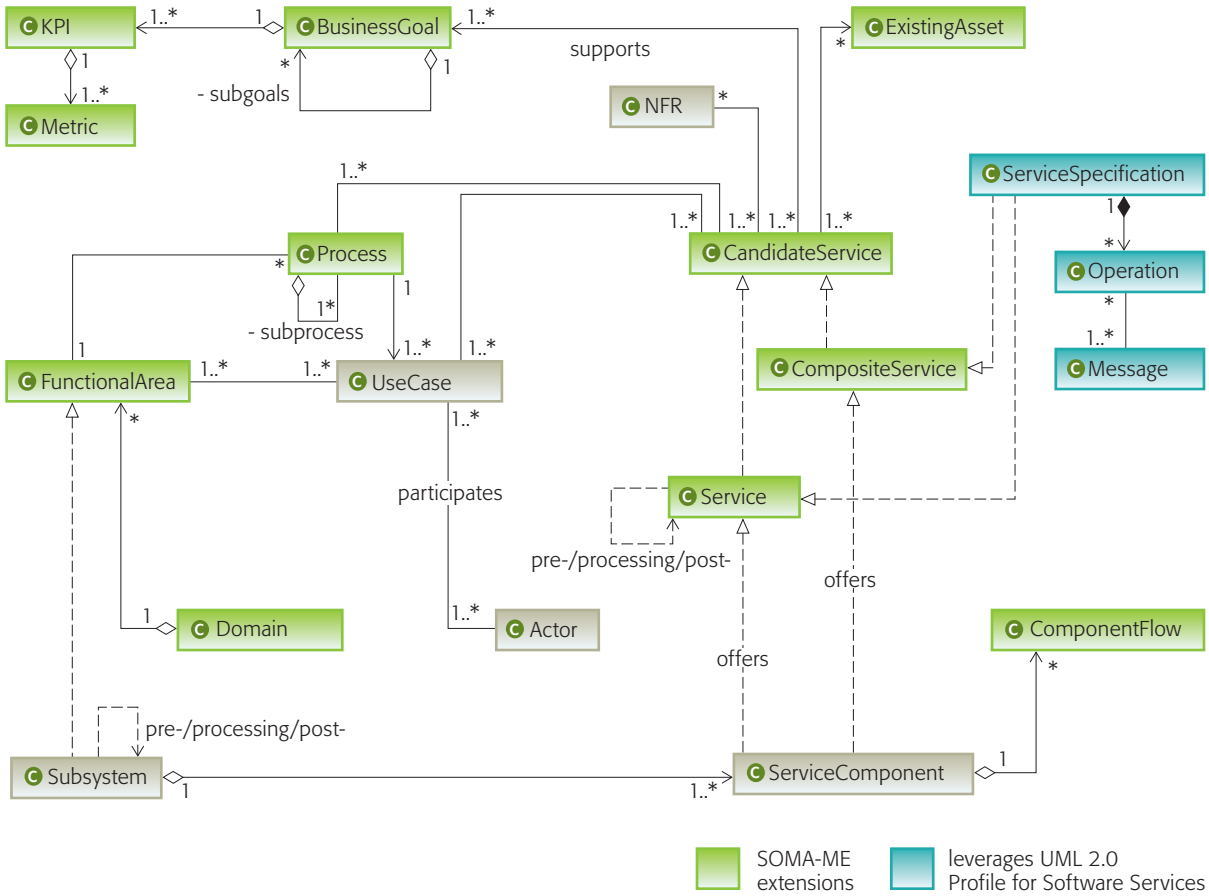


Figure 2
Metamodel for representing SOA solution artifacts

FPO

such as a particular service, component or subsystem. The appearance of an `<<ExistingAsset>>` model element depends on the type of Classifier to which the stereotype is being applied.

`<<Stereotype>> CandidateService` represents a candidate service discovered using various service identification techniques (such as process decomposition). The `id` and `description` attributes are associated with this stereotype. This information can then be incorporated into work products generated from the model (*work products* are the delivered documents on requirement analysis, business process definitions, and technical design and implementation). A set of four Boolean attributes: `eliminatesRedundancy`, `isBusinessAligned`, `isComposeable`, and `isExternalized`, are provided for recording the results of applying a selection algorithm to candidate services (an algorithm for selecting candidate services to be exposed, based on a number of predefined criteria). The technique attribute of the `<<CandidateService>>` stereotype can be used to specify the service identification technique(s) used. This attribute can hold multiple values, which are the encoded representation of the three techniques: domain decomposition, goal-service modeling, and existing assets analysis. The `exposureRationale` attribute can be used to record additional information to justify the exposure decision made for the candidate service.

`<<Stereotype>> CompositeService` represents a service that is obtained through the composition of multiple services. The state attribute is used to specify state management options for composite services, such as the status of a transaction, an invocation, and a security enforcement activity.

`<<Stereotype>> Service` represents a service that has been selected for exposure in the service

specification step. The nfr attribute can be used to specify nonfunctional requirements of the service. Formal requirement links can be used to map to nonfunctional requirements. The realizationType attribute can be used to specify a realization decision: Buy, Integrate, Subscribe, Build, Transform, or Outsource.

<<Stereotype>> CandidateServiceOrigin is used on a dependency link to indicate the origin of a particular candidate service in the service portfolio (e.g., the business process task, function of a functional area, or business goal from which it was identified). The originType attribute indicates the source of the services.

<<Stereotype>> SLTCategory represents a set of logically related criteria that are used to evaluate a candidate service in making exposure decisions. A category defines a weight attribute, which represents the relative importance of the criteria (compared with other defined categories) when determining exposure decisions. A scoringFunction attribute defines the formula for calculating the category score based on the values of the individual criteria in the category.

<<Stereotype>> SLTCriterion represents a quantitative measure that is used to evaluate a candidate service in making exposure decisions (made available for use by other applications or customers). A criterion defines minValue and maxValue attributes which represent the range of values that can be assigned for the criterion when evaluating each candidate service in the service portfolio.

<<Stereotype>> ExposureScope represents the extent to which services can be exposed (e.g., department or enterprise). The body of the ExposureScope defines the mathematical expression that is used to evaluate each candidate service based its assigned values for the SLTCriteria. The expression can include the SLTCriteria, numbers, and the following operators: +, -, *, and /. The threshold attribute defines a minimum score to be used as a cutoff point for selecting services to be exposed.

<<Stereotype>> ServiceComponent models a service component of a subsystem that realizes some selected services.

<<Stereotype>> FunctionalComponent models a functional component of a subsystem that provides business functionality.

<<Stereotype>> TechnicalComponent models a technical component of a subsystem that provides generic functionality.

<<Stereotype>> ComponentFlow models the interaction between service components in the system. This stereotype is attached to interaction diagrams describing the flow of messages between service components.

Figure 3 presents SOMA-ME as a framework that is independent of the development tools used. As an example, the SOA solution stack (S3)[21] is used to represent the layered and interconnected architecture of an SOA solution. The contents of the SOA solution stack are gradually filled in during the life cycle of the generic SOA design process. In order to build reusable SOA solutions, the model-driven architecture approach is leveraged to streamline the SOA solution design process.

As shown in Figure 3, S3 represents a generic SOA solution as a layered architecture. The S3 solution repository stores all the assets that are required to build an S3. The types of assets in the S3 solution repository include architectural building blocks (ABBs), KPIs, dependencies and interaction patterns, options and design decisions, method activities, S3 UML models, S3 modeling templates, and S3 industry-specific models. Utilities have been developed to add, remove, and modify assets in S3 solution repository. SOMA-ME supports asset exchange with other repositories, such as third-party SOA repositories, which is a desirable feature in a model-driven SOA solution design platform.

As an end-to-end SOA solution tooling framework with traceability features, SOMA-ME provides a cohesive assembly of methods, techniques, tools, and content in order to support the asset creation phase, from business modeling through the generation of technical documentation and code artifacts. SOMA-ME enables links between the solution artifacts (such as process, service, and service component) and existing industry-specific data structures. It is noted that the modeling templates for an SOA solution can be populated and custom-

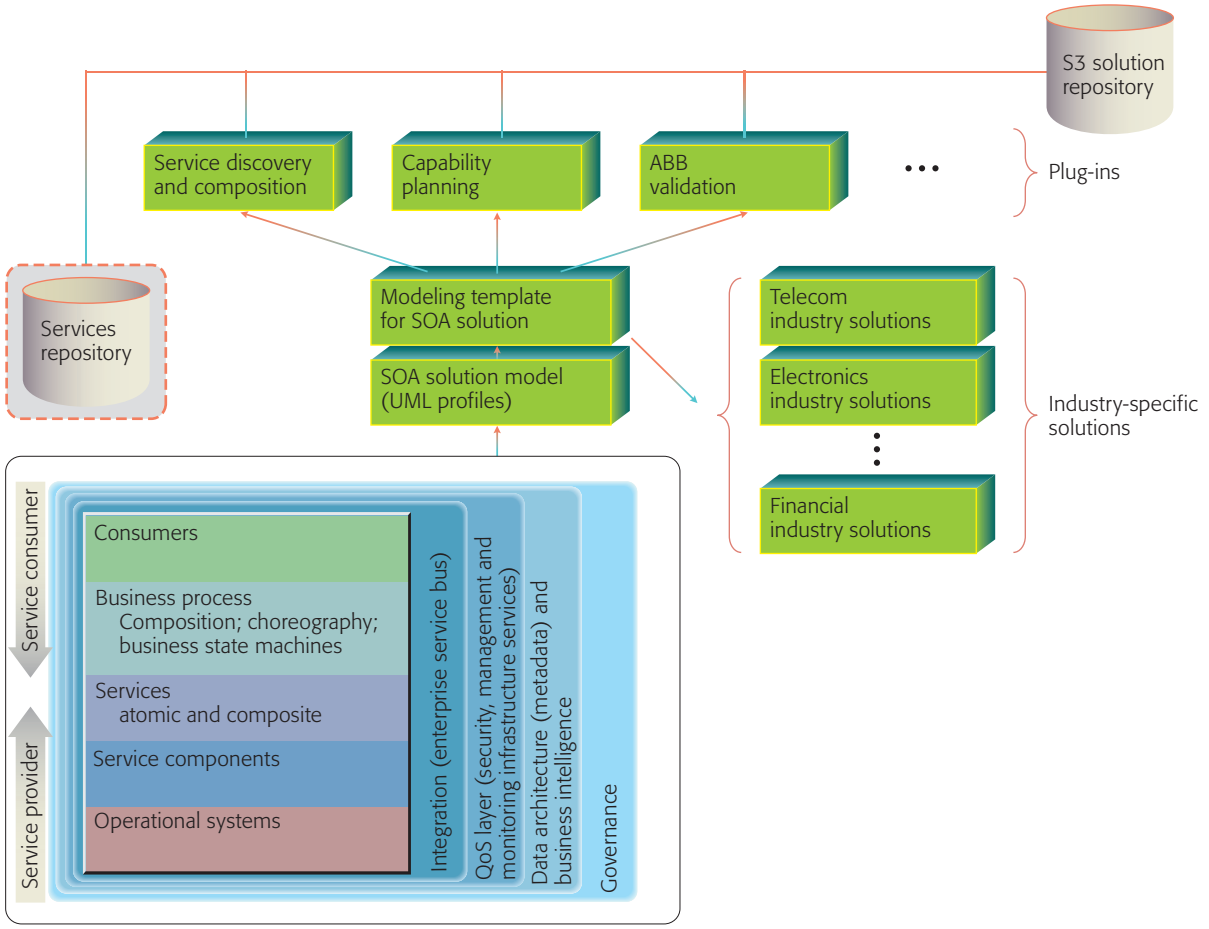


Figure 3
Key components of SOMA-ME

FPO

ized for specific industries, such as telecommunications, electronics, and finance (Figure 3).

In order to make SOMA-ME more flexible, a plug-in framework is provided. The following example plug-in utilities have been identified as key components in SOMA-ME: *service discovery and composition*, *capability planning*, and *ABB validation*.

The service discovery and composition utility plug-in is used for discovering Web services and composing Web services into business processes. It searches for services in multiple Web service registries in a federated manner with configurable aggregation capability. This utility can also translate solution requirements into XML-based search scripts to streamline the services discovery process. As an example, the IBM Business Explorer for Web Services (BE4WS) technology can play an important role to support federated services discovery in SOMA-ME.[9]

The service discovery and composition utility also supports globally coordinated composition of Web services flows (for example, composite services or business processes). It integrates the relationship profile in the S3 with the composition algorithm in order to adapt to changing constraints as well as business contracts. IBM Web Services Outsourcing Manager (WSOM)[7] is an example of optimization-based services composition framework that supports requirements-driven discovery and composition of Web services.

The capability planning utility is designed to use dynamic capability planning algorithms to coordinate the identification of IT services needed to realize the target business services. The capability

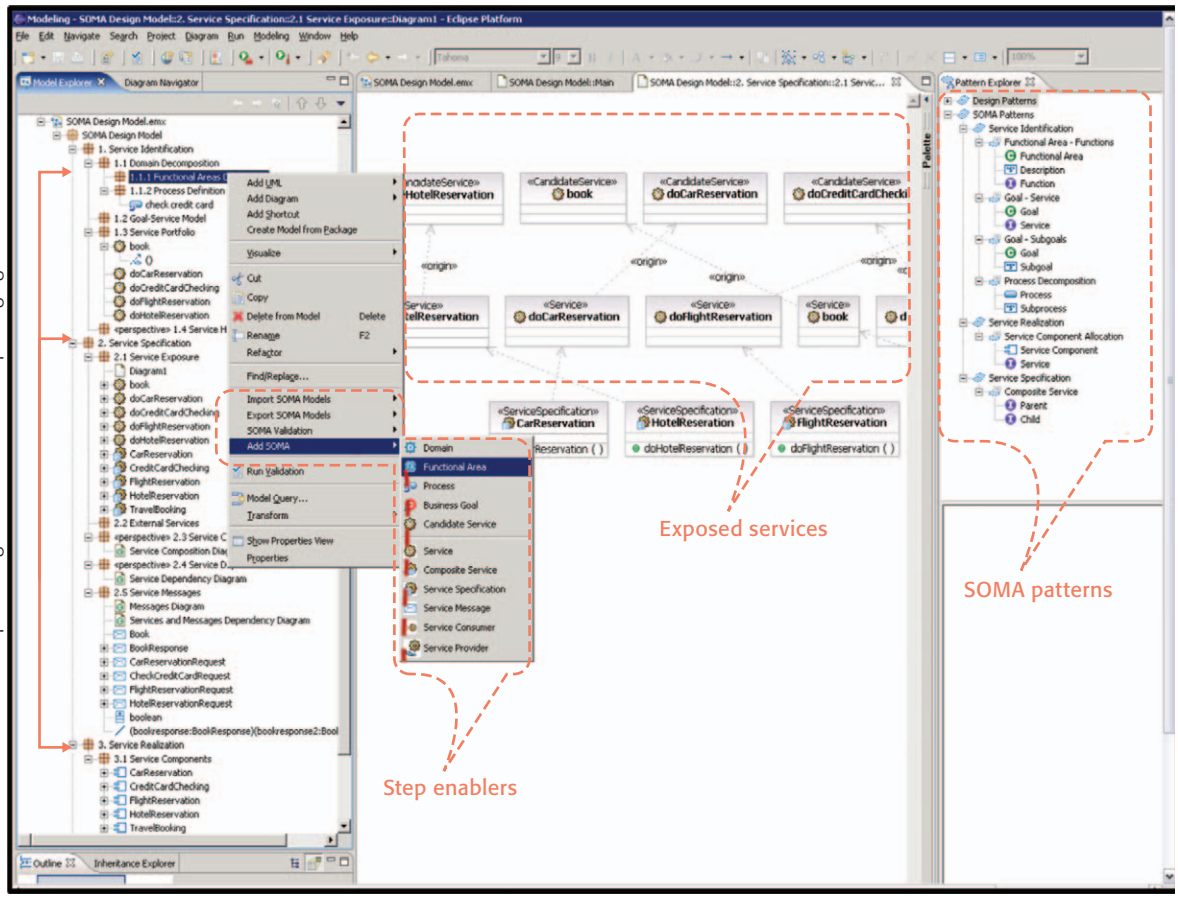


Figure 4
SOMA-ME user interface

FPO

planning utility also supports capability composition and decomposition based on existing IT system and application requirements. For example, existing software products and applications are used as inputs to explore the technical feasibility when we perform capability planning for a distributed SOA solution. It will be connected with the deployment aspect that handles the services ecosystem as well as individual SOA solution deployment for enterprise applications in the integration and management scenarios.

In SOMA-ME, the objective of an ABB validation algorithm is to automatically check for correctness and completeness, and resolve conflicts in order to ensure accurate relationship bindings of ABB for each layer in the S3. New validation algorithms can be introduced in this framework as long as they can support integration interfaces for ABB validation and analysis.

USING THE SOMA-ME TOOL TO DEVELOP SOA SOLUTIONS

Using SOMA to develop an SOA solution usually involves an IBM consultant working with a customer. Within IBM, SOMA is also used to develop reusable SOA solutions; that is, solutions that can be customized for multiple customer engagements. SOMA has a number of sequential steps, where the output of each step constitutes the input to the next step.[4,5] The SOMA-ME tool provides an integrated development environment that supports the use of SOMA by providing guidance throughout the design process and automation capabilities for some of the design steps. It is built as an extension to Rational Software Architect (RSA).[6]

Figure 4 illustrates the user interface of the SOMA-ME tool. The support for the SOMA method steps of service identification, service specification, and service realization is shown as the modeling

template on the left side of the figure. The key components of SOMA-ME are captured by the annotations shown in Figure 4.

In this section, the functions of the tool are presented and illustrated with examples. These include: context-aware guidance through the SOMA steps, reusable SOMA design patterns, a customizable algorithm for selecting the services to be exposed (to be used for generating other services by composition), modeling of the SOA solution architecture, a pattern-recognition-based facility for generating technical documentation and code artifacts, and achieving traceability through domain-specific impact analysis.

Context-aware guidance

The SOMA-ME tool provides UML profiles,[18] modeling templates of the SOMA service model, context-aware menus, SOMA patterns, and transformations that support the major steps of service-oriented identification, specification, and realization in the SOMA method.

The tool incorporates a GUI customization engine that can be used either by the user to customize the tool in the field or by the tool developer to extend its functions. This engine changes the tool development process from a *Java-coding* working style to a *text-editing* style, which lowers the required skill level for using the tool, thus making it accessible to a broader population. The GUI customization engine has been used to implement context-aware menus and domain-specific impact analysis and traceability features in the tool. Here, in the service identification phase of SOMA, a domain can be decomposed into multiple functional areas and a set of process definitions. With its context-aware menus, the tool helps to eliminate ambiguities in creating the SOMA services model.

To illustrate the use of the tool, here we show how to conduct the domain analysis step in a hypothetical banking application. In SOMA, the term *domain* is used to denote a logic grouping of business capabilities. After analyzing such a bank scenario, we separate the problem into four subproblems with separate domains.

1. *Traditional Banking*: corresponds to different types of banking services such as opening a checking account or a savings account
2. *Insurance*: corresponds to the insurance services such as automobile or home insurance
3. *Investing*: corresponds to the investment services such as education funds or retirement plans
4. *Loan*: corresponds to the loan services such as mortgages

Such identified domains can be added by selecting the directory labeled Functional Areas Description and using its context-aware menus.

Using similar context-aware menus, each domain can be further broken into several functional areas by identifying major functional responsibilities, which themselves can be separated into sub-functional areas even further. For example, the domain Bank may contain two functional areas, such as bank-side management and customer gateway.

Performing such add-domain operations continues until an appropriate level of business decomposition based on the available information is reached. Generally speaking, levels domain, functional area, and function comprise our hierarchical structure.

Reusable SOMA design patterns

In the SOMA-ME tool, SOMA patterns are created to capture reusable steps and solution artifacts. Examples of such patterns are service component allocation, goal-service modeling, and functional area analysis. For example, the "Service Messages" pattern is used to specify the input and output messages associated with a particular service. The "Composite Service" pattern is used to specify service dependencies between a parent service and the child services it invokes. The "Service Component Allocation" pattern is used to allocate services to a service component, which is used to implement or realize the corresponding services. The pattern can be selected in the Pattern Explorer in RSA, under the "Service Realization" group of "SOMA Patterns" shown in Figure 4.

In the banking example, we have a set of service components denoted as *customer access gateway*, *bank monitoring gateway* and *fund transfer component*. Customer access gateway provides services like "submit request," "contact bank associates," and "maintain personal account." The relationship between services and service components can be realized by "service component allocation pattern"

in SOMA-ME tool. Once the pattern is selected from the Pattern Explorer, the target model is selected, the services and the realizing component are selected or entered in a form-based dialog, and the pattern framework populates the corresponding elements in the model with their required stereotypes and relationships.

Algorithm for selecting services to be exposed

We investigated a services clustering algorithm based on pattern recognition theory in Reference 23, in order to categorize services based on business function. To each such cluster, we apply an algorithm for selecting the services to be exposed. Our algorithm is a criteria-based quantitative algorithm known as the *service litmus test* (SLT). The objective of SLT is to assist the user in making exposure decisions for the candidate services in the service portfolio. The SOMA-ME tool supports the definition of customized SLT criteria, which can be used when a quantitative approach is necessary in the process of making these exposure decisions.

A customized SLT consists of a number of criteria, each of which has a defined range of numeric values, that are used to evaluate each candidate service for exposure. Exposure scopes define the scope in which a service is to be exposed (e.g., departmental versus intra-enterprise versus inter-enterprise), and consist of a scoring function and a threshold for determining which candidate services should be exposed to the scope. The scoring function describes how the criteria are combined in order to compute a score for each candidate service. For example, in the context of a particular solution, a requirement for the purposes of making exposure decisions might be to rate each service on a scale of 1 to 5 in terms of: 1) its ease of implementation; 2) its value to the project at hand; and 3) its potential for reuse. This can be represented in the SOMA model using three `<<SLTCriterion>>` elements: `EaseOfImplementation`, `ValueToProject`, and `PotentialForReuse`. Each candidate service is then assigned a value for each of the three criteria.

In order to determine how to rank the services quantitatively, suppose that the value to the project of the candidate service is the main consideration, with implementation ease and reuse potential as secondary concerns. SOMA-ME allows the user to enter the scoring function and threshold for the exposure scope. The score for each candidate service can then be computed from the above weighting function, given the values that were assigned for each of the criteria. A threshold can then be assigned as a cutoff point for determining which candidate services should be exposed for further specification through SOMA. The SLT can then be conducted in the SLT view using the customized criteria by directly editing the values. As an example, the exposed services are illustrated in the middle of Figure 4.

SOA solution architecture modeling

S3 modeling support in the tool leverages the metamodel (in the SOA architecture-profile package) needed to model an SOA solution architecture. The SOMA-ME tool provides UML profiles, a modeling template, patterns, and transforms that support layered architecture, architectural building blocks, architectural decisions, and nonfunctional requirements for constructing SOA solutions or composite applications based on S3.[21] The S3 model in SOMA-ME provides a generic way of designing an SOA solution architecture in model-driven manner.

As shown in ***Figure 5***, the SOMA-ME tool has captured an initial set of layer-specific reusable architectural elements defined in S3 (S3 has nine layers, as shown in Figure 3). A modeling template for the S3 has been created to support SOA solution design. The results of the service model created from SOMA can be used in the business process layer, services layer, and services component layer in S3.

Figure 5 illustrates the user interface of the SOMA-ME tool for S3. The Model Explorer view shows the nine architecture layers of the modeling template provided by SOMA-ME. The user interface presents a consistent model structure for capturing the architecture. The package name and the location in the template determine the corresponding ABB allowed to be generated in this package. The tool supports establishing the relationship of ABBs based on their stereotypes. Property values of an ABB can be entered through directly editing the corresponding UML artifact or selected from a list of legitimate values that has been predefined in the tool.

Automating the generation of solution artifacts

SCA is being adopted as a programming model for implementing SOA-based components.[19] Based on the UML models captured in the SOMA-ME tool, SCA-compatible code segments that include WSDL,

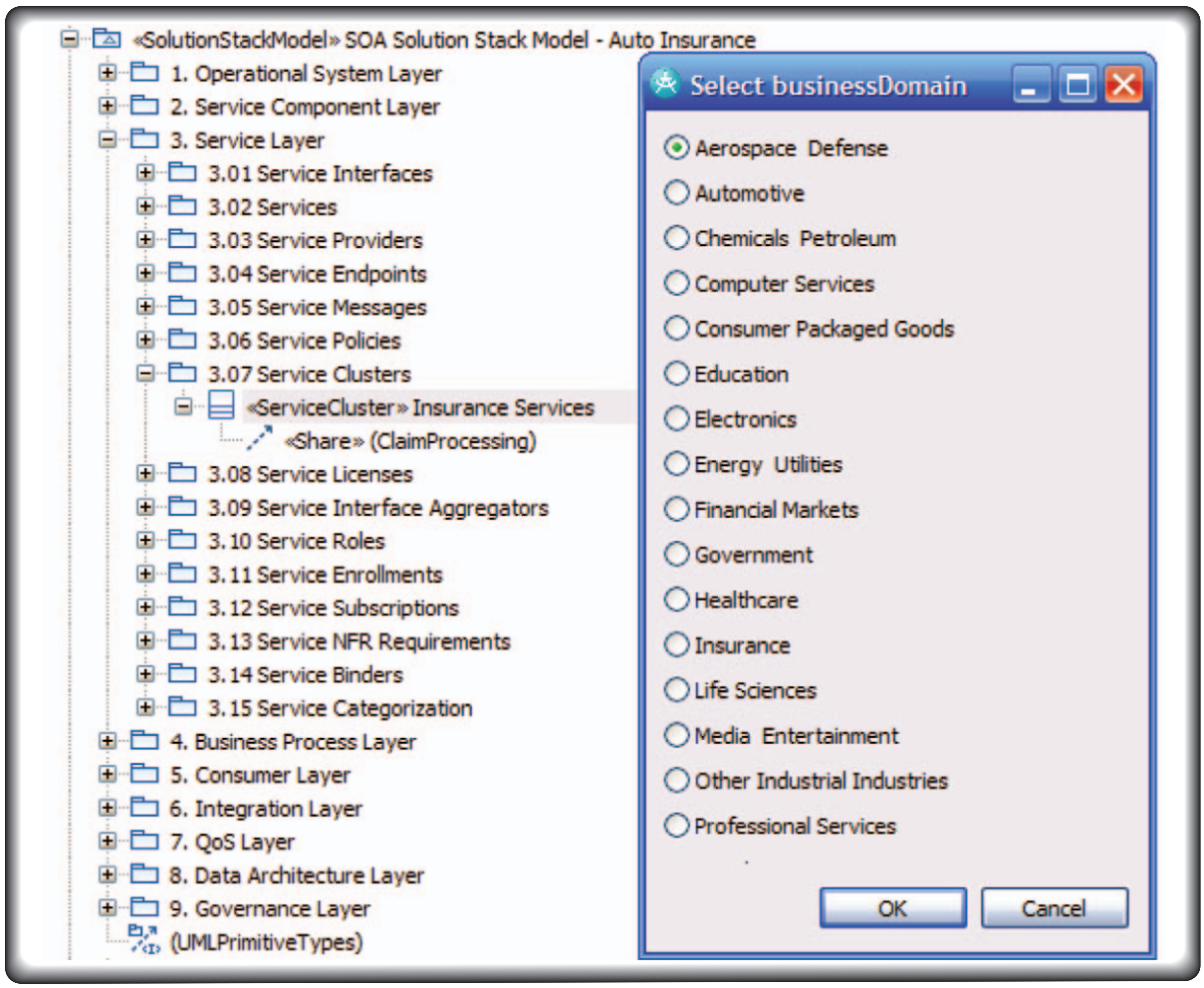


Figure 5
SOA solution modeling in the SOMA-ME tool

FPO

XML Schema, BPEL (Business Process Execution Language), and other SCA modules can be automatically generated from SOMA-ME. UML-to-SOA transformation support in RSA includes the following: UML to BPEL, UML to WSDL, and UML to XSD (XML Schema Definition). The generated SCA modules can then be opened in IBM WebSphere* Integration Developer (WID)[24] for subsequent development activities.

The SOMA-ME tool leverages the transformation support from RSA, but makes it completely transparent to the end user. When the modeling of the related set of solution artifacts with SOMA-ME tool is complete, SCA artifact generation can be initiated using a SOMA service model to SCA transformation. The transformation feature uses the SOMA service model contents to generate an intermediate RSA service model and then programmatically invokes the RSA platform support for SCA module generation.

In addition to this code-level solution artifacts generation, a set of documents including functional area description, service model, and component model can be delivered to a customer for an SOA project. Those documents can be in Microsoft Word, Excel,** or PowerPoint** format. The templates of those documents are created based on consulting and services delivery practices.

The SOMA-ME tool presents an extensible pattern-recognition-based artifact-generation framework to automatically extract data, including texts and

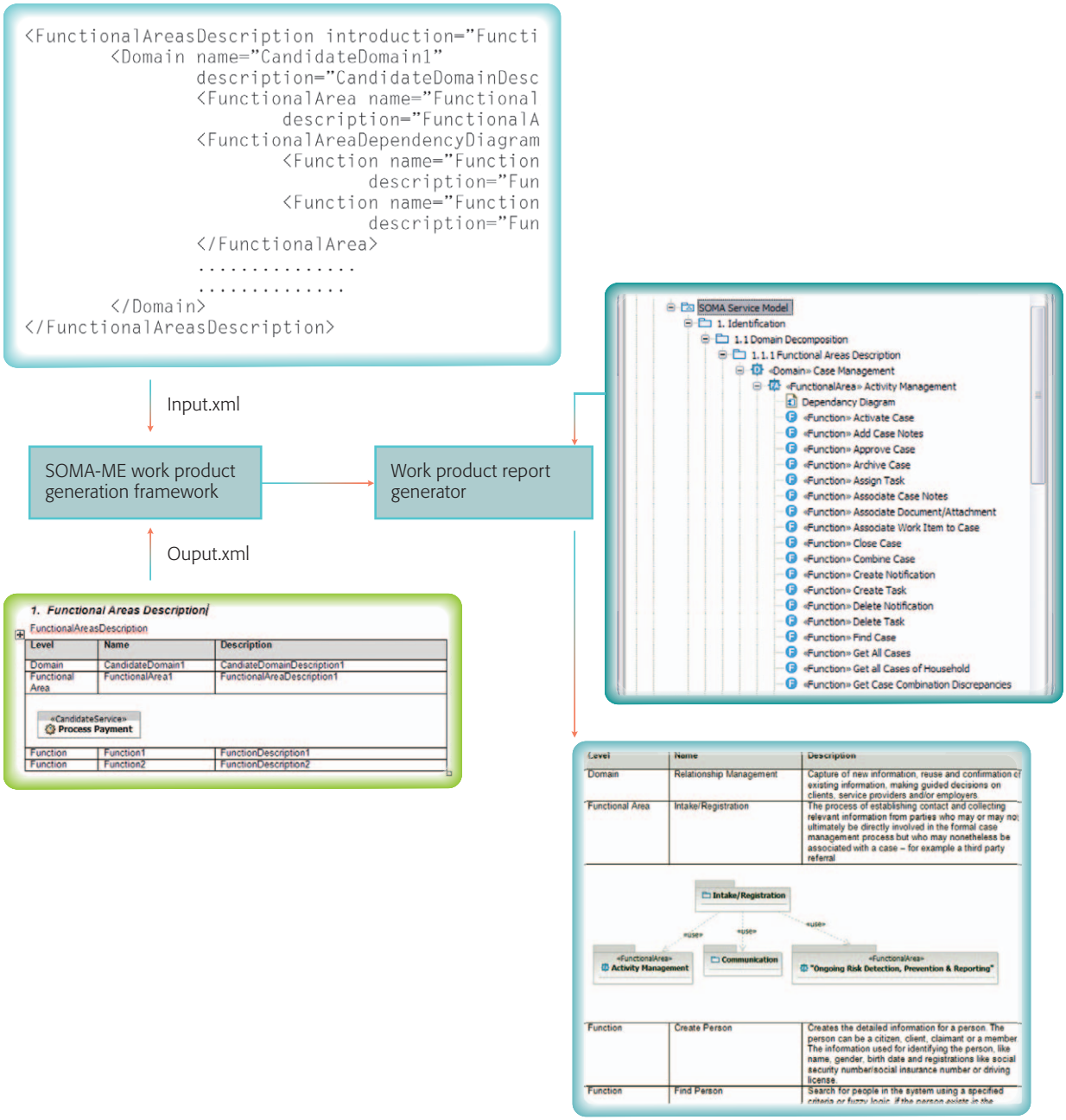


Figure 6
Work product customization and report generation

FPO

diagrams, from UML models and to automatically generate deliverable and editable documents (a.k.a. work products). In principle, an XML-to-XML transformation can be performed by using Extensible Stylesheet Language (XSL) transformations. In particular, many applications that produce formatted documents, such as Microsoft Word, PowerPoint, and Excel, now support an XML format. However, without in-depth knowledge of the target XML schema, it is difficult to construct an XSL transformation to the target XML. It remains a significant challenge to hand-code the transformations necessary to generate a high-quality target document due to the complexity of the target XML. For example, an existing approach includes the current document generation framework in RSA,

wherein an XSLT stylesheet needs to be manually created to transform the intermediate XML (in file model.xml) generated by the framework into a Word document in XML format.

The documentation generation component in the SOMA-ME tool allows for additional customization of the template used to generate the documents. This allows users to customize the appearance and content of the generated documents. Work product template customization consists of two parts: authoring the customized template, and importing the template into the modeling environment (i.e., the SOMA-ME tool) so that it is available for document generation.

Figure 6 depicts work product customization and report generation. Input.xml consists of model elements with predefined tags. The document template containing the same set of predefined tags and the required style is defined. The document generation component takes as input the two XML documents and automatically generates a Work Product Report Generator containing the XML-to-XML transformation. The Work Product Report Generator can be used to leverage the SOMA service model to generate the customized work products.

Traceability through domain-specific impact analysis

In the SOMA-ME tool, an initial version of traceability enablement has been implemented in the form of domain-specific impact analysis for the SOMA service model. The relationships among ABBs and other solution artifacts have been captured to align the design of the application architecture with business requirements and goals. A configuration file for impact analysis is created for defining the analysis starting point (a stereotype name) and ending point (another stereotype name) for the model. Furthermore, the relevant relationships between stereotypes are selected using the same configuration file. By using this configuration file, by setting the starting point, the ending point, and the relationships among the artifacts of the model, SOMA-ME filters out certain parts of the model.

To accommodate the needs of different scopes of analysis for a SOMA-ME model, we create two different types of impact and completion analysis.

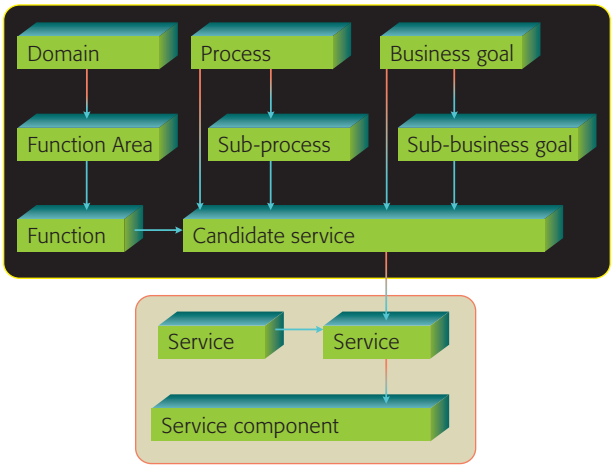


Figure 7
Stereotypes and relationships used for end-to-end impact analysis

FPO

The first one (End-to-End Impact Analysis) is an end-to-end model analysis, where the starting stereotypes are the legitimate starting points defined by the SOMA method, which are: `<<Domain>>`, `<<BusinessGoal>>`, `<<Process>>`, and `<<ExistingAsset>>`. The ending point of the end-to-end impact and completion analysis is `<<ServiceComponent>>`. The second analysis (Identification Phase Impact Analysis) only focuses on the service identification phase of the modeling effort; here, we keep the same starting points as for end-to-end analysis, but the ending point of the analysis is the `<<CandidateService>>` stereotype.

The impact analysis is configurable. The configuration file contains: 1) the stereotypes to be analyzed; 2) the selections of the starting points of the analysis; and 3) the selections of the ending points of the analysis. The remaining part (the main body) of the configuration file defines the relationships needed to complete the impact analysis diagrams. *Figure 7* shows the stereotypes and relationships used for this version of SOMA-ME analysis.

For a given impact analysis, two impact diagrams will be created if the analysis is valid for the selected artifact. They are:

- *Direct impact*—Defined as the diagram of a given artifact with other directly impacted (offspring) artifacts (usually, these artifacts become invalid when the given artifact is deleted)

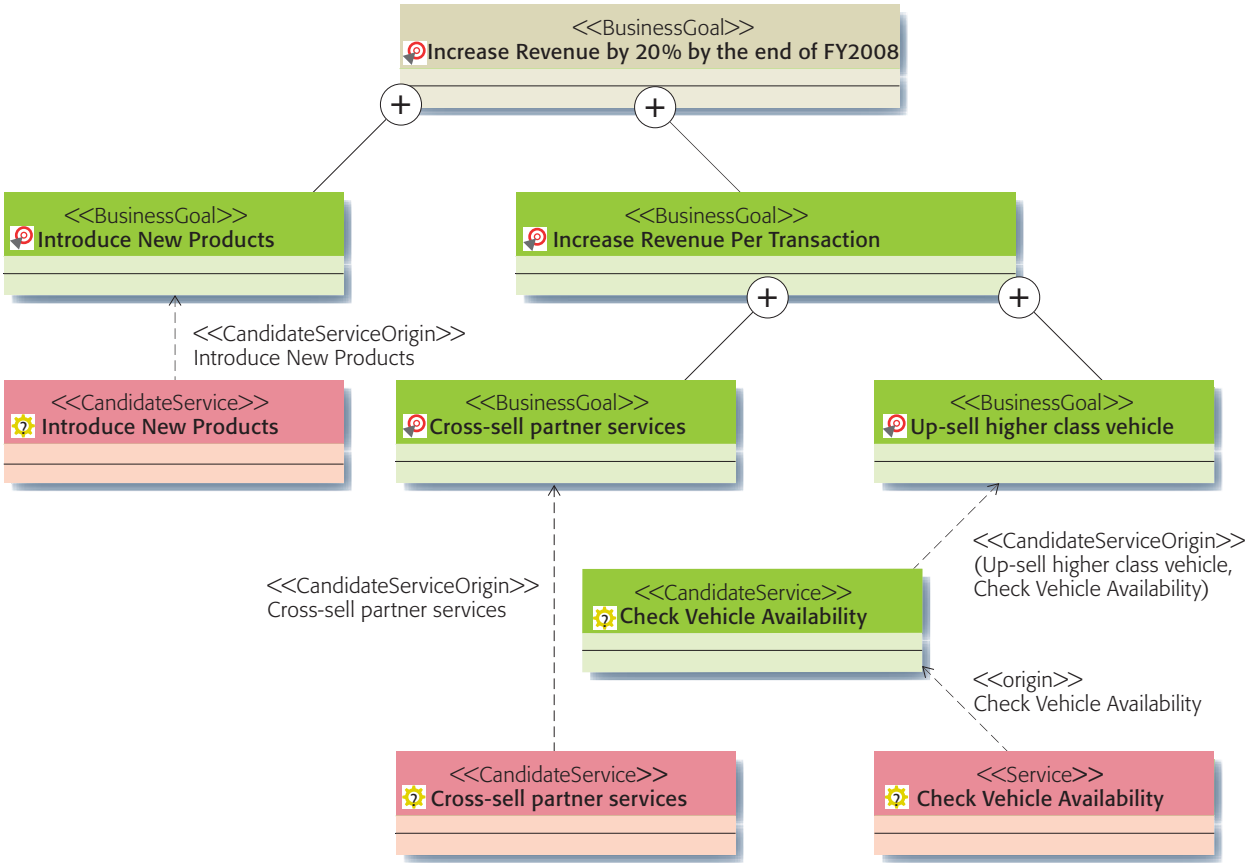


Figure 8
Example of direct impact diagram for artifact <<BusinessGoal>>

FPO

- *Indirect impact*—Defined as the diagram of a given artifact with other indirectly impacted (ancestor) artifacts (usually, these artifacts might be still valid in the model)

The completion analysis is accomplished by using color coding. For a direct impact, the leaf node has to have the ending point stereotype defined in the configuration file. If not, the leaf node will be marked as pink to alert the practitioner that the model is not completed. For an indirect impact analysis, the root nodes have to have one of the stereotypes defined as the starting point. Otherwise, a pink color will be used to mark these root nodes without proper stereotypes to alert the practitioner.

Figure 8 provides an example of a direct impact diagram for a <<BusinessGoal>> artifact. These pink artifacts reflect the completion status of the model. An artifact colored pink implies that this part of the model is not complete. For the case illustrated in the figure, any business goal should have one <<CandidateService>> to associate with, or more than one.

It is noted that the same impact analysis technique can be used for the variation-oriented analysis[18] in an SOA solution design.

Usage scenarios

In SOA practices, the SOMA method is used to gradually and iteratively populate the details of S3. The SOMA-ME tool helps to automate and guide SOA practitioners to design SOA solutions based on the SOMA method in a reusable and systematic way. The metadata and modeling template of the S3 is used as a generic solution template, which could be customized for various scenarios such as Composite Business Services (CBS),[25] Packaged Application Integration, and Custom Application Development. Two typical usage scenarios are briefly summarized below.

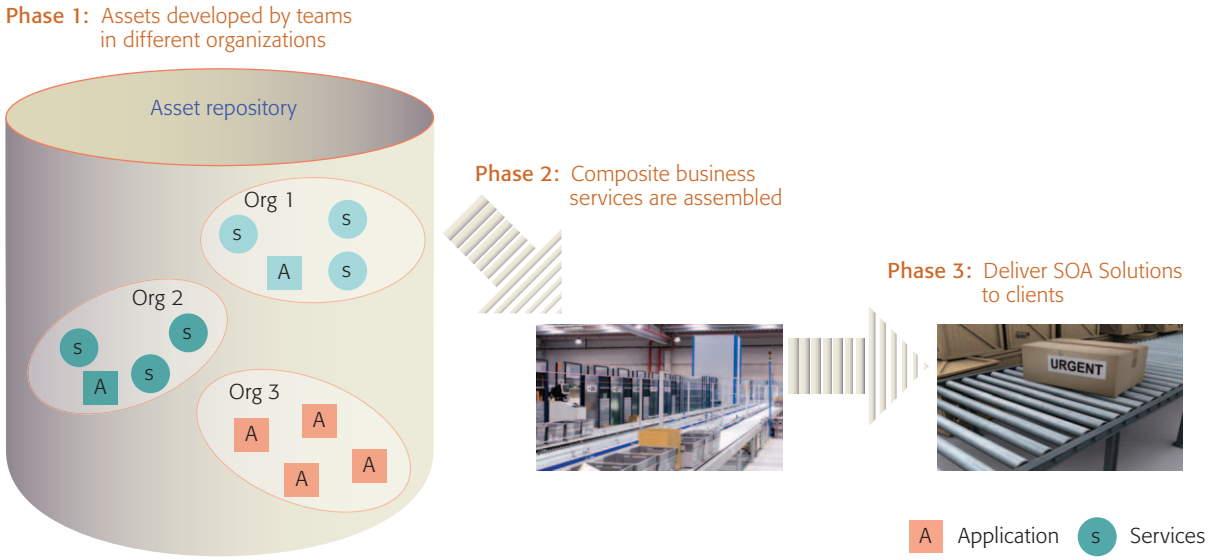


Figure 9
SOA manufacturing model

FPO

The design of the service model is a frequent usage scenario of the SOMA-ME tool. The service model is a key work product resulting from a SOMA engagement. This work product gets populated during various stages of SOMA. Other related work products (e.g., functional area description) are also produced during the service model creation. The SOMA-ME tool can be used to create service models in specific industries, such as insurance, retail, financial, government, and electronics.

For generic SOA solution design scenarios, we can use the S3 model to allocate the SOMA solution artifacts to the right layers and right ABBs in the S3. Then we can use the S3 model to perform architecture design review and gap analysis as well as other value-added services.

In summary, SOMA-ME has been created to help SOA practitioners design and create SOA solutions and services in a configurable and systematic way. The SOMA-ME tool helps streamline the SOA solution design process as an extension of the IBM Rational tooling platform to support the SOMA method and S3.

DISCUSSION AND CONCLUSION

Any SOA solution design process, and SOMA in particular, can be customized and enhanced for different industry domains or solution scenarios. In any such design process, the SOMA-ME model is populated with information specific to the SOA solution. Once solution modeling is complete, the model can be used to generate design documents, code artifacts, and other work products in an automated fashion, all of which can be captured in Word format.

The SOMA-ME framework we describe in this paper and its associated tool can be viewed as a manufacturing model of SOA solutions. This manufacturing model relies on a governed environment comprising tools, processes, architectures, and repositories, which are used to build reusable assets in a systematic way. ***Figure 9*** illustrates the SOA-solution manufacturing model as an "assembly line" process consisting of three phases.

As shown in Figure 9, in Phase 1 assets are created. Assets, which include applications and services, are developed by teams in various organizations. Applications are labeled A, services are labeled B, organizations are labeled Org 1, Org 2, and Org 3. In Phase 2 composite business services are generated through the assembly of various applications and reusable services. This operation is conducted by collaborating teams at multiple sites. In Phase 3 supporting materials for the business services created in Phase 2 are produced: installation manuals, user guides, and case studies. In the last

phase services are delivered to clients and maintenance services are provided as needed. All phases are coordinated and executed in an SOA solutions and services assembly line shown in Figure 9. The critical enabler of the SOA manufacturing model is a streamlined, end-to-end tooling platform that supports the development of the solution components. It is noted that some reusable artifacts could be placed in the asset repository for further use.

It is noted that some of the governance processes can be integrated with generic SOA solution development methods such as SOMA from IBM. However, this paper focuses on the tooling aspect of the SOA solution design, which can be applied or adapted to any similar SOA methods an enterprise may select.

SOMA-ME is an end-to-end SOA solution design framework that supports the model-driven design of SOA solutions using the SOMA method. The SOMA-ME tool, an integrated development environment built on the IBM Rational software development platform, helps IT architects evaluate and design an SOA solution architecture that conforms to S3, the IBM SOA reference architecture. The tool also supports model validation, impact analysis, variation propagation analysis, XML-based artifact generation, and model-centered visualization.

Future efforts will be directed toward "analytic SOA," an optimized SOA design approach based on a rigorous mathematical foundation. The quality of the SOA solution will be taken into consideration early in the design process and used to guide the entire solution life cycle. The analytic data will be associated with the solution artifacts in order to optimize the selection and composition of solution patterns and architectural building blocks at three levels: enterprise consulting, solution design, and infrastructure deployment.

ACKNOWLEDGMENTS

We thank Dingding Lu, Shuxing Cheng, Jia Zhang, and Krishna Ratakonda for their contribution to the work described here and Dr. Jarir Chaar, Ralph Nelson, Sridhar Iyengar, Dr. Robert Morris, and Dr. George Galambos for their support. We also thank the technical review board members Ray Harishankar, Kerrie Holley, Al Hamid, Shankar S. Kalyana, Teresa Hamid, Jerry Haegele, Dr. Joe Huchel, Abdul Allam, Dr. Min Luo, Siddharth Purohit, and Scott M. Simmons. Special thanks to the solution architects who have used SOMA-ME in client engagements. The excellent comments we received from the three anonymous reviewers have helped improve the quality of this paper.

CITED REFERENCES

1. *Service-Oriented Architecture—SOA*, IBM Corporation, http://www-306.ibm.com/software/solutions/soa/.
2. Special Issue on Service-Oriented Architecture, *IBM Systems Journal* **44**, No. 4 (2005).
3. L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*, Springer and Tsinghua University Press, New York and Beijing (2007).
4. A. Arsanjani, "Service-oriented Modeling and Architecture: How to Identify, Specify, and Realize Services for Your SOA," IBM developerWorks (2004), http://www.ibm.com/developerworks/library/ws-soa-design1/.
5. *IBM RUP for Service-Oriented Modeling and Architecture V2.4*, IBM developerWorks, November14, 2006, http://www.ibm.com/developerworks/rational/downloads/06/rmc_soma/.
6. *Rational Software Architect*, IBM Corporation, http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html.
7. *Web Services Outsourcing Manager*, alphaWorks, IBM Corporation (2002), http://www.alphaworks.ibm.com/tech/WSOM.
8. L.-J. Zhang and B. Li, "Requirements Driven Dynamic Services Composition for Web Services and Grid Solutions," *Journal of Grid Computing* **2**, No. 2, pp. 121–140 (2004).
9. L.-J. Zhang, T. Chao, H.-Y. Chang, and H. Li, *Business Explorer for Web Services*, alphaWorks, IBM Corporation (2001), http://www.alphaworks.ibm.com/tech/be4ws.
10. L.-J. Zhang, T. Chao, H. Chang, and J.-Y. Chung, "XML-Based Advanced UDDI Search Mechanism for B2B Integration," *Electronic Commerce Research* **3**, No. 1–2, 25–42 (2003).
11. H. Jain, H. Zhao, and N.R. Chinta, "A Spanning Tree Based Approach to Identifying Web Services," *International Journal of Web Services Research* **1**, No. 1, 1–20 (2004).
12. A. D'Ambrogio, "A Model-driven WSDL Extension for Describing the QoS of Web Services," *Proceedings of the 2006 IEEE International Conference on Web Services (ICWS 2006)*, Chicago (2006), pp. 789–796.
13. R. Grønmo, D. Skogan, I. Solheim, and J. Oldevik, "Model-driven Web Service Development," *International Journal of Web Services Research* **1**, No. 4, 1–13 (2004).
14. G. Ortiz and J. Hernández, "Toward UML Profiles for Web Services and their Extra-Functional Properties,"

Proceedings of the 2006 IEEE International Conference on Web Services (ICWS 2006), Chicago (2006), pp. 889–892.

15. R. Barrett and C. Pahl, "Model Driven Design of Distribution Patterns for Web Service Compositions," *Proceedings of the 2006 IEEE International Conference on Web Services (ICWS 2006)*, Chicago (2006), pp. 887–888.
16. J. H. Hausmann, R. Heckel, and M. Lohmann, "Model-Based Development of Web Services Descriptions Enabling a Precise Matching Concept," *International Journal of Web Services Research* **2**, No. 2, 67–84 (2005).
17. A. Kozlenkov, G. Spanoudakis, A. Zisman, V. Fasoulas, and F. Sanchez, "Architecture-Driven Service Discovery for Service Centric Systems," *International Journal of Web Services Research* **4**, No. 2, 82–113 (2007).
18. L.-J. Zhang, A. Arsanjani, A. Allam, D. Lu, and Y.-M. Chee, "Variation-Oriented Analysis for SOA Solution Design," *Proceedings of the 2007 IEEE International Conference on Services Computing (SCC 2007)*, Salt Lake City (2007), pp. 560–568.
19. BEA Systems, et al., *Service Component Architecture*, IBM developerWorks, November 2006 (updated November 2006), http://www.ibm.com/developerworks/library/specification/ws-sca/.
20. *Unified Modeling Language (UML)*, Object Management Group, Inc., http://www.uml.org.
21. A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah, "S3: A Service-Oriented Reference Architecture," *IT Professional* **9**, No. 3, 10–17 (2007).
22. S. Johnston, *UML 2.0 Profile for Software Services*, IBM developerWorks (April 2005), http://www.ibm.com/developerworks/rational/library/05/419_soa/.
23. L.-J. Zhang, S. Cheng, Y.-M. Chee, A. Allam, and Q. Zhou, "Pattern Recognition Based Adaptive Categorization Technique and Solution for Services Selection," *Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference (APSCC 2007)*, Tsukuba Science City, Japan (2007), pp. 535–543.
24. *WebSphere Integration Developer*, IBM Corporation, http://www-306.ibm.com/software/integration/wid/.
25. *WebSphere Business Services Fabric*, IBM Corporation, http://www-306.ibm.com/software/integration/wbsf/features/index.html.

Accepted for publication March 25, 2008.

Liang-Jie Zhang
IBM Thomas J. Watson Research Center, Hawthorne, NY 10532 (zhanglj@us.ibm.com). Dr. Zhang, a Research Staff Member and program manager of application architectures and realization in the Services Computing department, has been leading SOA projects, including the SOMA-ME project, for the past eight years. He has been coleading the SOA Solution Stack (a.k.a. SOA Reference Architecture) project since 2004. His book, *Services Computing*, was published by Springer in 2007. He has 37 patents and 20 pending patent applications. He is the founding Editor-in-Chief of the *IEEE Transactions on Services Computing* and the founding chair of the Professional Interest Community on services computing in the IBM Research Division.

Nianjun Zhou
IBM Thomas J. Watson Research Center, Hawthorne, NY 10532 (jzhou@us.ibm.com). Before joining the IBM Research Division, Dr. Zhou, a Research Staff Member in the Services Computing department, led the efforts in grid computing in the office of the IBM Chief Information Officer. Prior to joining IBM, he was Research Scientist in the New York State Department of Environmental Conservation. He received his Ph.D. in electrical engineering from Rensselaer Polytechnic Institute.

Yi-Min Chee
IBM Thomas J. Watson Research Center, Hawthorne, NY 10532 (ymchee@us.ibm.com). Mr. Chee, a Senior Software Engineer in the Services Computing department, has more than 15 years experience in software development tools and similar areas. His expertise includes tools for C++ programming environments, toolkits and standards for electronic ink-enabled applications, programming models and tools for high-performance computing on systems based on the Cell Broadband Engine™, and, most recently, model-driven design and development of SOA-based solutions.

Ahamed Jalaldeen
IBM Global Business Solution Center, Bangalore 560029, India. (ajalalde@in.ibm.com). Mr. Jalaldeen is an IT Architect working on methods for designing SOA solutions.

Karthikeyan Ponnalagu
IBM India Research Lab, Bangalore KA 560071, India (karthikeyan.ponnalagu@in.ibm.com). Mr. Ponnalagu has been working at IBM since 2000 and has been part of the India Research team since 2006. His research interests include separation of concerns, aspect-oriented programming, and MDA. He has authored two Redbooks in his IBM career in addition to other publications and has earned several patents. He earned an M.S degree in software engineering from Birla Institute of Technology.

Renuka R. Sindhgatta
IBM India Research Lab, Bangalore KA 560071, India (renuka.sr@in.ibm.com). Mrs. Sindhgatta is a Technology Staff Member working on artifact generation techniques and process change management for SOA solutions.

Ali Arsanjani
IBM Global Business Services, Fairfield IA 52556 (arsanjan@us.ibm.com). Dr. Arsanjani is a Distinguished Engineer and Chief Architect for the SOA and Web Services Center of Excellence within IBM Global Business Services, specializing in harvesting and developing best practices for the modeling, analysis, design, and implementation of SOA and Web services. He leads the company-wide SOA and Web services Community of Practice (comprising more than 5000 members) and is the principal author of the SOMA method for SOA.

Fausto Bernardini
IBM Thomas J. Watson Research Center, Hawthorne, NY 10532 (fausto@us.ibm.com). Dr. Bernardini, a Senior Manager in the Application Services Technologies department, leads worldwide research efforts to develop new technologies and methods for improving the efficiency and value of application development and management services. His current research focuses on SOA design methodologies, enterprise architecture modeling, and project management. Previously he held research staff, manager, and strategist positions in the IBM Research Division. He holds 13 patents and has published more than 50 scientific papers. He holds a Laurea degree in electrical engineering from the University of Rome, Italy, and a Ph.D. degree in computer science from Purdue University. ■

EIC Editorial: Introduction to the Body of Knowledge Areas of Services Computing

Liang-Jie (LJ) Zhang, *Senior Member*, *IEEE*

IN the inaugural issue of the *IEEE Transactions on Services Computing* (*TSC*), I used "SOA," "service-oriented consulting methodologies," and "services delivery platform and methodology" as examples to introduce the multi-level structure of the body of knowledge areas of Services Computing. Since body of knowledge areas can help the readers, authors, reviewers, editors, and community members at large to get a landscape view of the emerging Services Computing discipline, I would like to take this opportunity to introduce the structure and relationships of the details of the newly published *TSC* taxonomy [3]. *TSC* uses those key knowledge areas as the foundation to create a disciplined approach to organizing paper submissions, reviewers, and editors.

In order to have a comprehensive introduction to all the identified knowledge areas, I will extract and reorganize some contents from [1], [2]. Just to recap from [1], the 14 main knowledge areas of the Services Computing discipline included in the *TSC* taxonomy [3] can be categorized into the following four categories: Category 1 is about Services and Services Systems, which includes Principle of Services (M1) and Services Lifecycle (M2); Category 2 is about Services Technologies, which includes Web Services (M3), Service-Oriented Architecture (M4), Services Relationships (M5), Services Composition (M6), and Business Process Management & Integration (M7); Category 3 is about Services Consulting and Delivery, which includes Business Grid and Cloud Computing (M8), Enterprise Modeling and Management (M9), Service-Oriented Consulting Methodology (M10), and Services Delivery Platform and Methodology (M11); and Category 4 is about Services Solutioning and Management, which includes Application Services and Standards (M12), Security, Privacy, and Trust in Services Computing (M13), and Services Management (M14).

1 Services and Services System

In this category, key concepts on services and lifecycle are introduced. From a modeling perspective, a dynamic view of services has been put into the context of systems.

1.1 Principle of Services (M.1)

The first-level knowledge area, Principle of Services (M.1), includes general topics that cover subareas such as Services Systems, Services Models, Services Technologies, Services Architectures, and Optimization of Services Systems. "Services represent a type of relationships-based interactions (activities) between at least one service provider and one service consumer to achieve a certain business goal or solution objective" [2]. In the area of Services Computing, Services Systems are mainly implemented by IT-based software systems that realize a certain business service. In general, each system has at least one input and at least one output. In order to make a system dynamic, a feedback control framework can be used to take sensors' inputs, process the collected information, and take actions to achieve its business objectives. Services Models cover all aspects of mathematical modeling and model-driven developments. Traditional system modeling approaches can be applied to services systems through enhancements or customization. New service-oriented technologies are covered in the area of Services Technologies. The typical service-oriented technologies include Service Oriented Architecture, Web services, and Grid and Cloud Computing, which will be addressed in Category 2. Services Architectures are related to any architectural aspects at both business and IT levels. The connection between business architecture and IT architecture is one of the major focus areas of Services Computing.

1.2 Services Lifecycle (M.2)

Services Lifecycle (M.2) includes general topics, Key Factors in Services Lifecycle, and Service-Oriented Business Models. The general topics of services lifecycle cover the following six phases: Consulting and Strategic Planning, Services Engagement, Services Delivery, Services Operation, Services Billing, and Services Management. In general, Consulting and Strategic Planning is the first step to get socialized with clients and help identify pain points and strategic directions through consulting services. For example, a decision of outsourcing part of the business operations to a third party could be made in this phase. Once the client would like to start building a solution to address the identified challenges, a Services Engagement process will get started to get the memory of understanding or contract signed. The Services Delivery phase is responsible for analyzing the captured business requirements, designing a solution, and then delivering the solution for the client by working with its assigned team. Once the service is delivered to client, the client will move to the Services Operation phase to deliver services to its own customers on a daily basis.

For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org.

The operation process could be done by the client or a dedicated service operation team from other companies. In order to get the reward from delivering a service, Services Billing is a very important step to charge the usage of services based on some pricing strategies. When regular monitoring requests or exceptions happen, Services Management is a critical phase to make sure the deployed service can be consumed based on the original service level agreements. It is noted again, each phase of the services lifecycle can be performed by one or multiple parties.

The second key knowledge area under the umbrella of services lifecycle is the Key Factors in Services Lifecycle, which covers Data and Information, Processes, People, Resources, Financial Factors, Knowledge and Skills, as well as Innovation and Technology. Data and Information represent the inputs, outputs, and dynamics of services systems. Processes are a set of activities for achieving certain business objectives. In general, processes are repeatable for others to follow. People are central to a services system. People can drive a services delivery process or part of a delivery process that leverages people's decisions. Resources are physical resources, such as an office building, IT resources, such as a data server, information resources, such as electronic documents and presentation files, or abstract resources, such as time. Financial Factors are obviously important in terms of cost, unit price, and profit. Knowledge and Skills can directly influence the productivity and quality of services delivery in the current labor-intensive services businesses. In the services modernization stage, people's knowledge and skills can be captured and realized in reusable software systems to become secret weapons for the practitioners in the services industry. Innovation and Technology are the driving forces of growing and modernizing services business. New technologies can bring more opportunities for business model innovation.

The third key knowledge area of services lifecycle is Service-Oriented Business Models. The innovation of business models can help grow a business from a profit perspective. Service-oriented componentization technology and reuse-based composition and assembly can help create and realize new business models in a more efficient and effective manner. Actually, creating or adopting a new business model may be easy. But, the execution of a business model for the best performance is hard. In the area of Service-Oriented Business Models, Services Modernization, Software as a Service, and Services as Software are three major approaches. Services Modernization is used to transform the operations of a services business to be enabled or automated by the latest information technology. One example of services modernization is to streamline the supply chain management for retail services. Services as Software and Software as a Service (SaaS) represent new services delivery models that deliver and operate values added services around a software stack. For instance, exposing a customer relationship management (CRM) software application as a service for a wide range of users over the Internet is a typical SaaS business model. Services as Software is a new model of transforming the currently used labor-intensive services business model to an asset-based services business model. The key idea is to realize the best practices from the field in reusable software tools to increase the productivity and effectiveness of the services delivery process.

2 Services Technologies

2.1 Web Services (M.3)

As a realization technology of a service-oriented architecture, Web Services (M.3) have gone through an adoption phase to become a mainstream standard of defining interfaces and communication protocols for modularized components. Although Web services were initially introduced and adopted in business-to-business integration scenarios, they have been gradually used in various solutions in horizontal and vertical industry spaces. In addition to the general topics, the key knowledge areas of Web services include Composite Services, Web Services Publishing, and Web Services Discovery. The general topics of Web services include all aspects of the representation of Web services. Those knowledge areas include Web Services Modeling, Web Services Communication Protocols, Web Services Binding, Web Services Registry, Stateful Web Services, and Web Services Interoperability. Web Services Modeling covers the interface description languages or mathematical model for representing individual Web services. For example, the Web Services Description Language (WSDL) is an example of interface description languages. Web Services Communication Protocols covers how the communication is enabled between Web services. An example communication protocol is the Simple Object Access Protocol (SOAP) and its variations. The Web Services Registry covers the metadata, access protocols, and integration interfaces of repositories for publishing and discovering Web services. Web Services Binding covers how the invocation gets done. For example, we can use the SOAP Servlet to handle SOAP requests on a Web server. The Stateful Web Services area covers how to annotate and manage state information for a Web service. The Web Services Resource Framework (WSRF) is an example way of capturing the state information of a Web service. Web Services Interoperability addresses the integration and cross-platform reuse of Web services which are produced by various tools from different vendors. For example, WS-I Profiles for various scenarios can help achieve the interoperability of Web services.

The Composite Services knowledge area includes three subknowledge areas: Composite Web Services, Representation of Composite Services, and Multidimensional Modeling. Here, the knowledge area of Composite Web Services includes scenarios of getting multiple Web services to be part of a composite service, which defines its services interface based on individual Web services. Representation of Composite Services deals with how a composite service is described in a readable way. For example, BPEL can be used to describe a composite Web service. The knowledge area of Multidimensional Modeling extends the traditional WSDL-based static modeling to dynamic modeling, which takes time dimension and Quality of Services (QoS) into consideration, and further to relationship modeling, which captures various relationships among services and service providers.

Web Services Publishing is emphasized here to include knowledge areas for publishing services to services registries, which can be a Public Services Registry, Private Services Registry, and Distributed Services Registry. For example, UDDI registry could be used as a public service registry or private service registry. For Distributed Services Registry, Web Services Inspection Language (WSIL) documents or customized Really Simple Syndication (RSS) files could be used as services registries to announce new services or features of existing services. So, in this knowledge area, services registries and repositories are the major target places for Web services publishing.

The knowledge area of Web Services Discovery includes the Services Discovery Language, Services Discovery Engine, Services Discovery Process and Methodology, Service Discovery Architecture, and Federated Services Discovery. Here, Services Discovery Language is a direction of defining an effective and efficient query model and description language to support search for services. We can manually discover services based on keywords or semantic annotations. The Services Discovery Engine covers how a processing capability can be modeled and enabled to parse search queries and handle the search results for appropriate output generation. The Services Discovery Process and Methodology includes reusable steps that can help construct search query, resolve query conflicts, dispatch search requests, aggregate results, disseminate notifications, and present search results in appropriate formats. A well-formulated methodology of services discovery can help practitioners find the right services to be part of a solution in an effective and disciplined approach. The Service Discovery Architecture provides an enablement and realization framework of services discovery in a modularized manner. This architecture supports search requests from program clients and Web browser clients. A caching mechanism and parameters are part of the architecture, which also supports searching centralized services repositories or distributed documents (e.g., XML files). Federated Services Discovery is the most used services discovery scenario for creating a solution which leverages services from multiple services registries. The business case is equivalent to the scenario of an Internet search from multiple Web sites but in the service-oriented environment.

2.2 Service-Oriented Architecture (M.4)

Service-Oriented Architecture (SOA) includes five second-level knowledge areas: General Topics, Services Invocation, Bridging Business and IT Architecture, Solution Lifecycle, and Solution Reference Architecture. As a loosely coupled architectural framework, SOA provides an extensible and reusable approach to designing a solution. In the knowledge area of general topics of SOA, Operational Model and Realization aspects are covered. The conceptual SOA includes service consumer, service provider, and service registry to provide an interaction environment for consuming, publishing, and discovering services for achieving certain goals. The Operational Model includes practical ways of customizing or formalizing SOA in the context of solution design. The Realization area covers all implementation aspects from technology foundation to product offerings. For example, Web services are one of the multiple technologies that can be used to realize an SOA. Web 2.0 technologies can be also used to realize and implement user-centric SOA solutions.

Services Invocation is a major knowledge area of consuming services in a disciplined approach. It covers Simple Services Invocation, Metadata of Services Interfaces, Metadata Publishing, and the Advanced Services Invocation Framework. The area of Simple Services Invocation includes how a method signature (i.e., operations names and associated input and output parameter types) is constructed manually for invoking a Web service. An example is to analyze a WSDL file and manually construct a service invocation call. The meanings of the parameters and other semantics are not covered in the Simple Services Invocation area. The Metadata of Services Interfaces area covers how and what metadata should be captured to annotate additional information for method signatures of services. For example, MetaWSDL is one way of annotating additional information that is not captured in WSDL. MetaWSDL and WSDL can pair with each other to provide rich information for the potential automation of services invocation. Metadata Publishing specifically addresses how the metadata about interfaces can be associated with the existing interface description languages, such as WSDL and BPEL. For example, WSIL can be used to pair WSDL with the corresponding MetaWSDL in a hyperlinked fashion. Once the additional annotation is added to an existing service interface description file, the Advanced Services Invocation Framework can help construct method signatures of services for consuming services automatically or programmatically.

Bridging Business and IT Architecture is a key area of using SOA principles to capture and propagate business-level requirements and strategies to the process execution level and then to the IT realization level. It includes Enterprise Level Transformation, Process Level Transformation, and Programming Level Transformation. The Enterprise Level Transformation mainly concentrates on the componentization and prioritization of key business functions in an enterprise for transforming its current business to a to-be state that could achieve better and more effective business performance. In general, the output of the Enterprise-Level Transformation includes identified pain points and high-level strategies to launch transformation initiatives. The area of Process-Level Transformation addresses how to formalize, create, or reengineer a set of business processes to improve the business areas which were indentified in the phase of Enterprise Level Transformation. Then, Programming-Level Transformation can realize and implement the identified business processes through software development. Programming languages or models are part of the Programming-Level Transformation for SOA. The research challenges not only include the modeling of each transformation at all the three levels, but also cover how the gaps can be connected between levels.

Solution Lifecycle includes Solution Modeling, Solution Development, Solution Deployment, Solution Publishing, Solution Discovery, Solution Invocation, Solution Composition, Collaborations in Solution, Solution Monitoring, and Solution Management. Solution Modeling covers requirement gathering and analysis as well as architecture design. Solution Development includes the technology and tools selection for implementing a solution based on the

solution architecture from the solution modeling phase. Once a solution is developed and tested, it goes to the target environment for the Solution Deployment, which includes the installation and configuration of code, documentation, other solution artifacts (e.g., business processes), as well as corresponding software and hardware. Solution Publishing covers how a solution can be exposed as a service or application for others to leverage. It also covers repository and registry for enabling a solution or solution artifacts to be reused in the future. Solution Invocation includes how a solution can be leveraged. For example, a solution can be exposed as a service, which can be invoked like an individual service. On the other hand, if multiple services will be invoked, they should provide a centralized service invocation engine in the context of a solution for effective logging. Solution Composition covers how several solutions can be composed to become a composite application that provides an integrated solution view and other value-added services. Composition patterns and methodology can be major research topics from an engineering innovation perspective. The area of Collaborations in Solution covers how services are coordinated in a solution. For example, service-to-service collaboration protocols and enforcement mechanisms are research directions that may lead to OSI-like solutioning protocols for SOA. Solution Monitoring includes how key performance indicators are defined, tracked, and presented in a user-friendly or machine-readable format. Solution Management concentrates on the routine maintenance and exception handling when a solution is in operation mode.

The Solution Reference Architecture area includes various aspects of designing a solution architecture for SOA. It covers the Architecture Overview Diagram, User Interaction and Presentation, Processes, Services, Services Components, Operational Systems, Integration, Quality of Services, Data Architecture, and Governance. The knowledge area of the Architecture Overview Diagram provides a graphic diagram to get all key architectural elements to be illustrated. It may have several perspectives that are associated with corresponding audience types. The main audiences of the enterprise view of an architecture overview diagram are business analysts or senior solution architects. The IT view of an architecture overview diagram could target solution architects and developers. User Interaction and Presentation covers how a solution interacts with end users and programs. For example, it includes the architectural building blocks of modeling the interactions of a Web user or a business-to-business program with other architectural elements in a solution. The knowledge area of Processes deals with the design of all the business processes and their collaboration patterns in a solution. The Services aspect in solution architecture covers all the reusable services, their relationships, and their source and target solution artifacts. The Services Components area discusses how services are realized and implemented in components, such as Java class or .Net components. The knowledge area of Operational Systems includes all hardware, middleware, and application portfolios that support the deployment of services components and other architectural elements. The knowledge area of Integration deals with how applications can be integrated into a solution and how a message can be routed and shared among other solution artifacts and business entities involved in a solution. The Quality of Services area covers what quality metrics should be defined and applied to the right solution artifacts in a solution. The knowledge area of Data Architecture includes how various data formats can be integrated and transformed based on access control policies. The Governance area in a solution reference architecture handles organizational issues, design-time and run-time monitoring, and management based on best practices. For example, team building and information dissemination practices can help create a dedicated organization to focus on the innovations of SOA and related projects. The other type of governance includes a set of normative guidance that guides the practitioners to design solution architecture in a disciplined approach.

2.3 Services Relationships (M.5)

The knowledge area of Services Relationships covers general topics, the Web Services Relationship Language, and Service-Oriented Relationship Modeling. Some general topics include Relationships in Services Registries and Relationship Specification Languages. Since services are eventually cataloged in services registries, the annotation or metadata about relationships of services are important building blocks in a service registry. The other aspect is how to represent the relationship in a readable manner.

The knowledge area of the Web Services Relationship Language covers how to represent the relationship in a unified way. For example, an XML-based structure is used to capture relationships for Web services. Since Web services are provided by a business entity, the relationships can be defined at the business entity level, the Web service level, or the operations level.

Service-Oriented Relationship Modeling extends the Web Services Relationship Language to cover all service-oriented scenarios. It covers the relationships among business entities, business services, Web services, and operations. For example, an alliance relationship between business entities could provide better integrated business services offerings at lower prices. This kind of relationship can directly affect the selection of service providers and their corresponding services. This subknowledge area concentrates on the metadata of service-oriented relationship modeling. It covers Business Services Relationships, Modeling at the Business Entity Level, Modeling at the Business Service Level, and Relationship-Enriched Services Registry. The knowledge area of Business Services Relationship includes all kinds of relationships between business services, their providers, and their realizations. Modeling at the Business Entity Level includes four types of business-to-business relationships: partnership, parent-child relationship, exclusion, and alliance. XML schema can be used to represent those relationships. The knowledge area of Modeling at the Business Service Level includes relationship modeling of business services within one business entity or cross-business entities. For example, the following relationships can be captured to cover within and across business entities: parent-child, exclusion, binding, and community. Relationship-Enriched Services Registry is an engineering innovation area for enhancing services registries with well-defined and structured relationships.

2.4 Services Composition (M.6)

Services Composition includes general topics, the Services Integration Framework, and Services Value Chain Collaboration. It spans services integration within an enterprise or across enterprises. Services composition is a way of creating value-added services or applications based on existing services.

Some general topics of services composition include Aspects of Business Requirements, Business Requirements Modeling, Requirements-Driven Services Discovery, and Formalization of Services Composition. The knowledge area of Aspects of Business Requirements includes target components (e.g., user interfaces, functions, data models, events, and messages), operational environments, asset lifecycle governance, project management consideration, and finance consideration (e.g., development cost and price). Business Requirements Modeling covers ways of representing requirements in a readable manner. For example, XML can be used to capture all aspects of business requirements for services composition. The knowledge area of Requirements-Driven Services Discovery handles services discovery based on requirements. In most cases, automatic services discovery are based on modeled requirements. Once the services are indentified through requirement analysis and discovery, how to use a mathematical model to formulate services composition and present the composite services are the main topics of the Formalization of Services Composition area.

The knowledge area of the Services Integration Framework includes the Services Integration Procedure and Optimization of Services Composition. The Services Integration Procedure area explores methodologies of integrating various services into a composite service or service-oriented business process based on requirements. The knowledge area of Optimization of Services Composition covers how the resulting business process can be tuned to satisfy business requirements in a manual or automatic way. For example, a global optimization algorithm can be used to compose an optimal or near optimal business process based on requirements. Some best practices and performance data analysis of integrating services are also covered in this area.

The Services Value Chain Collaboration area covers how business collaborations are conducted in business-to-business environments. In this case, an enterprise is not standalone anymore. It collaborates with other partners to form a value chain based on its business objectives. Within an enterprise, the collaboration among multiple organizations or business units is also folded in this knowledge area. Therefore, the Services Value Chain Collaboration area includes Interenterprise Collaboration and Intraenterprise Collaboration. The other topics in this area are the Extended Business Collaboration Model, Annotated Business HyperChain, Web Services Collaboration Resources, Collaborative Message Primitives, Collaborative Constructs, and Collaborative Exchange Protocols. The knowledge area of the Extended Business Collaboration Model includes all models that support business collaborations across organization boundaries. The Annotated Business HyperChain area defines how the value chain can be represented without changing individual entities and their resources involved in the value chain. The knowledge area of Web Services Collaboration Resources defines how various resources (e.g., Site, Organization, Project, Task, People, Message, Transaction, Document, Product, and Tool) can be represented in a unified way. In this case, the Web Services Resource Framework (WSRF) could be used to capture all static or dynamic resources in a value chain. The Collaborative Message Primitives area covers reusable unit-level request-type or response-type primitives in a service-oriented environment. The knowledge area of Collaborative Constructs includes a set of reusable message exchange patterns that have specific business goals. Each collaborative construct is comprised of one or several message primitives. The Collaborative Exchange Protocols area covers how to leverage unified resources, message primitives, and collaboration constructs to annotate collaborative business processes that are involved in multiple organizations. A business solution can be composed by one or multiple collaborative exchange protocols.

2.5 Business Process Management and Integration (M.7)

The knowledge area of Business Process Management and Integration includes three second-level knowledge areas: general topics, Service-Oriented Business Process Management, and Flexible Business Process Integration. The general topics include all aspects of Business Process Modeling and Business Process Management. The first one covers how a business process can be represented in description languages (e.g., XML), graphic tools, or model-driven templates (e.g., UML). The later one includes how to monitor and manage business processes.

The Service-Oriented Business Process Management area includes Top-Down Process Management, Bottom-Up Process Management, Business Process Reengineering, and Process Reengineering Methodology. In the knowledge area of Top-Down Process Management, a business process is decomposed into subprocesses. A subprocess can be further decomposed into smaller subsubprocesses, until a task in the process can be realized by services. In general, Top-Down Process Management is mainly used in business driven and service-orientation scenarios. The Bottom-Up Process Management area deals with how to transform a legacy application into service-oriented business processes. In practice, Top-Down Process Management and Bottom-up Process Management can be used jointly at any point in time. The knowledge area of Business Process Reengineering covers how a business process is reengineered from the functional requirements perspective. For example, one task could be realized by two services in a reengineered process due to availability issues. The Process Reengineering Methodology area concentrates on the normative guidance of reengineering a process. For example, in an SOA solution design, the following three steps are used widely: process partition or decomposition, services allocation for realizing tasks in process, and realization of services through services components. All those major activities in the process reengineering methodology have reusable and configurable patterns that can be further articulated and analyzed based on theoretical models and best practices. An example method of process reengineering covers process decomposition, business services clustering, service selection, data

modeling, service definition, business logic refinement, service implementation, service deployment, service monitoring and management, and service maintenance.

The knowledge area of Flexible Business Process Integration covers the Lifecycle of an Integration Activity, Integration Activity Modeling, and Business Process Monitoring. In real business scenarios, various applications are required to be integrated with the main solution. But for each application, the integration interfaces and adapters are different. How can we increase the flexibility of business process integration? In order to address this challenge, an integration ontology can be created. Once a new application is to be integrated with the current solution, a new integration activity is captured in the integration ontology. Then, an adaption layer for this new integration activity needs to be implemented. Since all the adaption layers' service interfaces have the same definition of the input and output parameter types, only the contents in the input parameter have the context-aware information about this new integration activity. Examples of Flexible Business Process Integration can be found in [2]. The knowledge area of Integration Activity Modeling covers the modeling principles and theoretical analysis of deadlock free, for example. The Resource Description Framework (RDF) is one example way of representing the integration activity ontology. Here, the Business Process Monitoring area concentrates on how to prove a model of process integration is deadlock free. Formal verification techniques can contribute to this effort. For instance, Petri nets are used to model an adaptive activity management system to help simulate, analyze, and validate some important properties of a system from deadlock-free and safeness perspectives.

3 Services Consulting and Delivery

3.1 Business Grid and Cloud Computing (M.8)

The knowledge area of Business Grid and Cloud Computing includes general topics, Logical Grid Infrastructure, and Business Grid Solution Development. Some general topics include Service-Oriented Grid Computing, the Business Grid Solution Framework, and Cloud Computing. Grid Computing promotes resource sharing. Once resources can be represented as standard-based interfaces such as Web services, Grid Computing needs to address its service-oriented behaviors. The knowledge area of the Business Grid Solution Framework covers how to realize business resources and applications sharing in a layered Grid infrastructure, which includes the Physical Grid and the Logical Grid. The Physical Grid part can support physical resource sharing. The Logical Grid part supports business application sharing and business process sharing. Cloud Computing evolves resource sharing through the latest virtualization technology. As a result, Cloud Computing enables large-scale data centers to share their spare resources, such as storage and application hosting environments, with end users, which include enterprise users and individual users. In practice, Grid Computing mainly concentrates on computing resources rather than application and business process sharing. Even for today, parallel computing, clustering computing, and supercomputing are still the dominating areas in the Grid Computing community. Cloud Computing is a good business model for using virtualization technology to deliver business services. In the rest of this section, "Business Grid" can be used interchangeablely with "Business Cloud" because both of them deliver business value over a distributed network. However, the enabling technologies might be different.

The Logical Grid Infrastructure area deals with the infrastructure aspect of a logical Grid, which is a conceptual and abstracted Grid architecture. It includes the Packaged Application Grid, Business Grid Middleware, and Business Process Grid. The knowledge area of the Packaged Application Grid is to hide the complexity of existing applications and provide an interface layer as services. The Business Grid Middleware area covers how to provide an IT infrastructure and software middleware to support business applications in a Grid Computing environment. Business Process Grid area includes business process provisioning, outsourcing, integration, collaboration, monitoring, and management.

The knowledge area of Business Grid Solution Development includes Business Grid Service Development and Business Grid Service Invocation. Business Grid Service Development deals with how to design and develop services for the distributed Grid Computing or Cloud Computing platform. On the other hand, the Business Grid Service Invocation area concentrates on the consumption of the services in the Grid or Cloud Computing environment.

3.2 Enterprise Modeling and Management (M.9)

The Enterprise Modeling and Management area includes three second-level knowledge areas: general topics, Methodologies for Enterprise Modeling, and Enterprise Performance Management. The general topics of Enterprise Modeling and Management cover the Dynamics of Services Ecosystem and Requirements for Enterprise Modeling. The knowledge area of the Dynamics of Services Ecosystem concentrates on how new technologies such as SOA can be used to build a dynamic services system to adapt to changing business models. The area of Requirements for Enterprise Modeling deals with various stakeholders and different visibility requirements while modeling an enterprise.

The knowledge area of Methodologies for Enterprise Modeling includes Balanced Scorecard and Strategy Map, Component Business Modeling Circle, and Enterprise Architecture and Transformation. Here, the Balanced Scorecard and Strategy Map area provides a communication tool and continuous execution tool for modeling an enterprise. The corresponding perspectives of objectives, measurement, target, and initiative can be used to define strategy map. The knowledge area of the Component Business Modeling Circle takes business components as inputs to perform business analysis and produce high-level constructs for realizing business services.

The Enterprise Performance Management area includes Enterprise Project Management, Performance Management, Service-Oriented Enterprise Management, and Enterprise Portfolio Management. Enterprise Project Management

moves the traditional project management's scope to integrate business requirements, project data, project status, and resource information into one managed environment. Performance Management covers the performance tracking and management aspects of project-based businesses. The knowledge area of Performance Management covers idea formalization, initiative creation, and IT realization of the created initiatives. The Service-Oriented Enterprise Management area covers the componentization of enterprise management. An example service-oriented enterprise management system includes business objectives, projects (portfolio), tools, components, and corresponding actions which are performed by the involved organizations. The knowledge area of Enterprise Portfolio Management covers multiple projects in multiple categories at the project portfolio level. Step-by-step methodologies for enterprise portfolio management are also covered in this area.

3.3 Service-Oriented Consulting Methodology (M.10)

The knowledge area of Service-Oriented Consulting Methodology includes general topics and Service-Oriented Business Consulting. Some general topics in this area cover the Consulting Method for Strategic Change and the Consulting Method for IT Strategic Plan. When an enterprise becomes more complex in value chain settings, it needs more adaptive business models and supporting infrastructures to execute its growth strategy. A typical business and IT alignment method covers three aspects: enterprise level, process level, and IT infrastructure level. The knowledge area of the Consulting Method for Strategic Change includes strategy definition, governance model, and recommended process improvements. In this phase, the typical outputs are white papers that describe the market trends, gap analysis, and suggested improvement points. On the other hand, the Consulting Method for the IT Strategic Plan area covers the IT assessment phase, governance model defining phase, initiatives launching phase, and IT transition planning phase.

The Service-Oriented Business Consulting area deals with methodologies of conducting business consulting services in a service-oriented approach. The traditional consulting methods often ignore the possibility of reusing assets and leveraging open ecosystems. Some key activities in service-oriented business consulting methodologies are key topics in this area. They include Gap Analysis, Initiatives Identification, Value Chain Analysis, Portfolio Analysis, Transition Planning, Project Management, and IT Service Management. The knowledge area of Gap Analysis covers service-oriented analysis of the "AsIs" and "ToBe" status of an enterprise. The Initiatives Identification area includes how to identify business initiatives through a set of reasonable analysis steps. The identified initiatives are expected to align with the overall enterprise architecture, illustrate enhancements for specific business components in an enterprise, and lead the business to the target strategy. The knowledge area of Values Chain Analysis concentrates on the integration and management of business partners and components suppliers to maximize the business value in a distributed service environment. The Portfolio Analysis area includes topics such as project coordination, prioritization of each initiative, dependence analysis, and executive communication. The knowledge area of Transition Planning covers all detailed steps of enabling the identified initiatives based on the portfolio analysis results. The Project Management area concentrates on the execution of projects through resource identification, role clarification, budget management, work break down structure, and statement of work. The knowledge area of IT Service Management deals with how to ensure service operations to satisfy the predefined service level agreements.

3.4 Services Delivery Platform and Methodology (M.11)

Services Delivery Platform and Methodology includes general topics, Service-Oriented Services Delivery Platform, Services Delivery Methodology, Software as a Service, and Services as Software. The general topics of this area include Services Delivery Mechanisms and Services Engineering. The knowledge area of Services Delivery Mechanisms covers all possible delivery models and their evolutions. For example, most of the total or integrated services providers are moving to specialized services offerings based on their best practices in selected industries or domains. The Services Engineering area includes all engineering aspects of services delivery. It can also cover some aspects of services development and other phases of the whole services lifecycle if the services delivery is a continuation of previous services engagements.

The knowledge area of the Service-Oriented Services Delivery Platform includes the Traditional Services Delivery Platform, Collaborative Services Delivery Platform, and Common Services. The Services Delivery Platform area covers all required core infrastructure services, IT service management, horizontal services, vertical business services, service partnership management, value-added services, service membership management, and service lifecycle monitoring. The knowledge area of the Collaborative Services Delivery Platform concentrates on the collaborative view of the services delivery platform. It includes the business capability perspective, business-to-business collaboration perspective, and coordinated access control perspective. The Common Services area includes reusable services that can be consumed by other services or applications in the services delivery platform. For example, centralized membership management service, billing service, and search service are core services in a services delivery platform.

The Services Delivery Methodology area includes the following key steps: the Services Delivery Readiness Phase, Services Delivery Creation Phase, and Services Delivery Operation. The knowledge area of the Services Delivery Readiness Phase covers the interactions between the service provider and service consumer through a service method adoption phase that includes analyzing requirements, defining governance, creating information and data architecture, and enabling project management. The Services Delivery Creation Phase area includes management of service variation, implementation, testing, and deployment by leveraging the services delivery platform, technologies, and tools. The knowledge area of Services Delivery Operation includes incident management, change and problem management, release management, and configuration management.

The knowledge area of Software as a Service includes Web 2.0 and Web X.0, Service Mashup, and New Business Models. The key goal of Software as a Service is to deliver software applications as services in an efficient and effective way. Web 2.0 and Web X.0 cover all the latest and future technologies to enable services delivery over the Internet. For example, consumer-oriented applications can be enabled through rich interfaces realized by Web 2.0 technology. The Service Mashup area covers how services can be composited in a systematic way over the Internet. For instance, integrating map service with product sales information can generate a new value-added service in the context of service mashup. The knowledge area of New Business Models includes how to leverage Software as a Service to deliver brand new or improved service offerings. For example, moving a standalone product lifecycle management (PLM) tool to the Internet can attract more Web users.

The knowledge area of Services as Software includes the Asset-based Services Model and Services Software. The traditional service business mainly concentrates on the labor-intensive delivery model. The Asset-based Services Model is a way of transforming the labor-intensive service model to increase the productivity and quality of services delivery business. The knowledge area of Services Software includes how to use software to embed best practices from the field into reusable and consumable software tools or systems. It is expected that more and more software applications or tools will focus on the enablement of the knowledge accumulated from services engagements.

4 Services Solutioning and Management

4.1 Application Services and Standards (M.12)

The Application Services and Standards area includes five second-level knowledge areas: general topics, Solution-Level Quality of Service, Data Architecture Framework, QoS Management Modeling, Web Services Standard Stack, and Industry-Specific Standards. The general topics in this area include Case Studies in Industry, Case Studies in Scientific Applications, and Case Studies in Government. Those topics cover applications in industry sectors, scientific applications, and e-government solutions. Some cross-industry methodologies and tools are important enablers of asset reuse in multiple solution scenarios.

The knowledge area of Solution-Level Quality of Service includes the Context-Aware QoS Model, Representation of QoS Model, QoS Data Management, Business Relationship Model, and Solution-Level QoS Framework. The Context-Aware QoS Model area covers how to qualitatively and quantitatively define the quality of a solution. In general, the QoS model includes a set of attributes such as reliability, security, safety, and availability. The knowledge area of the Representation of QoS Model covers a uniform, flexible, and extensible way of modeling a QoS model for a solution. The QoS Data Management area covers how the QoS data is communicated and propagated through message exchanges including QoS metrics. The Business Relationship Model defines how service-oriented relationships can be leveraged to assure QoS for a solution. The knowledge area of the Solution-Level QoS Framework covers a logical QoS framework that controls and manages the quality of a solution. It may include multiple architectural building blocks in the enablement architecture.

The Data Architecture Framework area includes Constructs in Data Architecture, Relationships Between Constructs, and Data Services. The knowledge area of Constructs in Data Architecture includes data services gateway, data aggregator, data mining manager, access control manager, traceability enabler, data representation manager, and data sources manager. The Relationships between Constructs area covers some predefined relationships between constructs. For example, a one-to-one or many-to-many relationship between selected constructs can be defined. The relationships can be modeled in a class diagram in UML or in XML. The Data Services area concentrates on how to enable data as services.

The knowledge area of QoS Management Modeling includes Modeling of Resources and Modeling the QoS Assurance Process. The Modeling of Resources area covers modeling aspects of all entities involved in a solution lifecycle. The knowledge area of the Modeling QoS Assurance Process deals with how to provide reasonable assurance for a solution to get right QoS metrics and results in practice. Any description languages can be used to develop a model for a QoS assurance process. For example, BPEL could be used to capture a specific QoS assurance process.

The Web Services Standard Stack covers the following layers: Transport, Messaging, Description/Publishing/Discovery, Quality of Services, and Service Composition. SOAP, XML, WSDL, BPEL, UDDI, and Web services security are example standards in the Web services standard stack. The knowledge area of Industry-Specific Standards covers how to leverage the evolving SOA and Web services standards to define industry-specific solution frameworks, business processes, business protocols, and data models.

4.2 Security, Privacy, and Trust in Services (M.13)

The knowledge area of Security, Privacy, and Trust in Services includes four second-level knowledge areas: general topics, Access Control in Services Systems, Security Enablement in Services Systems, and Privacy Management in Services Systems. The general topics include Security Concerns of Service-Oriented Solutions, Privacy Concerns of Service-Oriented Solutions, and Trust in Service-Oriented Solutions. With the development of Internet technologies, more and more applications are moving to the Web. Who can access those applications? Has any private information about users been exposed to unauthorized users? Who can you trust on the Internet? Those questions and concerns are always associated with Web-based applications such as social networking solutions.

The Access Control in Services Systems area covers Role-based Access Control, Policy-based Access Control, Single-Sign-On, and Constraints and Rules. Role-based Access Control is used in a multirole collaboration environment to make sure only the right group of users can access the same type of information. The knowledge area of Policy-based Access Control covers how to define and enable policy to support access control. The Single-Sign-On area covers how to leverage role-based access control or policy-based access control mechanisms to get the right visibility with only one visible verification that needs user's manual interaction. Once the first verification process is done, the session will last for a predefined period without requiring another visible verification request to the end user. Constraints and Rules are used to define policies and actions for enabling security, privacy, and trust in a solution.

The knowledge area of Security Enablement in Services Systems includes Service-Oriented Security Enablement at the Software Level, Service-Oriented Security Enablement at the Hardware Level, Service-Oriented Security Enablement at the Solution Level, and Security Enablement Methods and Tools. Generally speaking, a solution includes hardware, software, and other supporting infrastructure. Security enablement can be embedded in a hardware component or a software component. However, from a solution perspective, the hardware-level security enablement and software-level security enablement need to be federated at the solution level. So, some additional security modules may be added to the solution to leverage the hardware and software-level security features. Tools and methodologies for enabling security can dramatically improve the consistency and productivity of the delivery process of security and privacy services.

The Privacy Management in Services Systems area covers Privacy Management in Data Collection, Privacy Management in Data Transformation, Privacy Management in Data Dissemination, and Privacy Governance Methods and Tools. Privacy has become a major concern in electronic business. When data is collected, the contributors of the data have concerns about the possibility of inappropriate exposure of the collected data. When data is transformed, the same privacy issue requires a governance process to prevent the involved parties from exposing data in an inappropriate manner during the data transformation. In the data dissemination process, there is a question of how to ensure that only the right receiver can get the privacy-related information. Therefore, systematic approaches should be summarized as delivery methodologies for privacy protection services. A set of corresponding tools can help privacy practitioners deliver regular privacy assurance service and serve as platforms for new innovations. For example, new privacy assurance verification and enforcement algorithms can be plugged into the existing tools to get high quality service delivery for privacy or security assurance.

4.3 IT Services Management (M.14)

The knowledge area of IT Services Management includes four second-level knowledge areas: general topics, Application Management in Services, Infrastructure Management in Services, and Business and IT Aligned Management Services. The general topics in this area cover Management of Services Design and Management of Services Delivery. Management of Services Design deals with how a service is developed in an organized fashion. For example, task assessment and resource allocation in a service project can be managed in various environments such as in-house development, cross-site development, or outsourcing. The whole services design process can be streamlined in a factory-like setting to produce services from a managed services assembly line. Management of Services Delivery area concentrates on the delivery aspects of a service. It includes people resource management, customer support, system maintenance, billing handling, delivery process improvement, and service quality monitoring and management. The goal of Management of Services Delivery is to use minimum resources to deliver high quality services to satisfy customers' requirements.

Application Management in Services includes three third-level knowledge areas: Application Management Services, Application Development Services, and Legacy Application Transformation Services. The Application Management Services area covers all kinds of maintenance services for existing applications which may need hardware management, software upgrade, application testing, and quality management or technical support service for the application users. The knowledge area of Application Development Services concentrates on the development aspects of applications. It involves governance design, development process improvement, business architecture design, application architecture design, software tools selection, middleware and implementation technology, code development, testing for application development, technical writing, and preparation of training and marketing materials. The Legacy Application Transformation Services area covers how a legacy application is migrated to a new solution architecture or integrated with other applications. For example, the legacy to SOA transformation effort has resulted in many code analysis algorithms and automation systems.

The knowledge area of Infrastructure Management in Services includes Maturity Assessment Services, Network Management Services, Server Management Services, and Data Center Management Services. The Maturity Assessment Services area deals with methods and tools of assessing the degree of maturity of the current IT infrastructure. As a result, a maturity assessment report can be generated. The knowledge area of Network Management Services covers the management aspects of network infrastructure, network software tools, network firmware, network configuration, security and price setup, priority management for applications over the network, and billing of network usages. The Server Management Services area includes server setup, clustering management, software upgrade, load balancing, service-level agreement monitoring for server's usages, operating systems support, and hosting support. The Data Center Management Services area concentrates on the management of storage, energy consumption management for Green compliance, storage outsourcing, data backup, and emergency handling services.

The Business and IT-Aligned Management Services area includes Service-Level Agreement for Contracts, Key Performance Indicators for Business Processes, Quality of Services for Services Offerings, and Management Methods and Tools for Business and IT Alignment.

5 Summary

It should be noted that all of the defined knowledge areas and subareas can be selectively composed to create various courses serving different audiences. In the last few years, the Technical Committee on Services Computing within the IEEE Computer Society has organized a series of tutorials, Summer School on Services Computing, Fall School on Services Computing, panels, the Education Methodology Summit on Services Computing, and planetary talks to get input from the participants of the IEEE International Conference on Web Services (ICWS), IEEE International Conference on Services Computing (SCC), Congress on Services (SERVICES), IEE International Computer Software and Applications Conference (COMPSAC), IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS), and IEEE Asia-Pacific Services Computing Conference (APSCC). I would like to send my special thanks to the organizing committee members of those events, the participants, and the *TSC* Editorial Board members for their contributions to this first version of the body of knowledge areas of Services Computing.

I hope the introduction of the body of knowledge areas of Services Computing inspires you to explore scientific foundations and killer applications in the field of Services Computing. Applying the Services Computing discipline to your daily teaching, consulting, research, development, or services delivery practices could enable you to be more effective and systematic in the process of modernizing the services industry. Case studies and course guidelines will be published in the Services Computing Curriculum handbook for degree programs.

6 About This Issue

I am very pleased to include four research papers in the second issue of *TSC*. All of the papers have gone through a comprehensive review process. The decisions were recommended by the assigned associate editors. The first paper is "A Secure Information Flow Architecture for Web Service Platforms" by Jinpeng Wei, Lenin Singaravelu, and Calton Pu. Wei et al.'s paper is in the knowledge areas of Web Services as well as Security, Privacy, and Trust in Services. The second paper is "Similarity-Based SOAP Multicast Protocol to Reduce Bandwidth and Latency in Web Services" by Khoi Anh Phan, Zahir Tari, and Peter Bertok. Phan et al.'s paper is in the knowledge area of Web Services. The third paper is "Dynamic Web Service Selection for Reliable Web Service Composition" by San-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee, and Cheng-Hung Chen. Hwang et al.'s paper is in the knowledge areas of Web Services and Services Composition. The fourth paper is "Coordinated Service Allocation through Flexible Reservation" by Kazuo Miyashita, Kazuyuki Masuda, and Fumitaka Higashitani. Miyashita et al.'s paper is in the knowledge area of IT Services Management.

I hope you like the papers published in this issue. If you have any suggestions or questions on *TSC*, please feel free to contact me so we can work together as ONE team to leverage *TSC* as a community-driven innovation platform for our Services Computing professionals.

Liang-Jie (LJ) Zhang
Editor-in-Chief

References

[1] L.-J. Zhang and C.K. Chang, "Towards Services Computing Curriculum," *Congress on Services—Part I (SERVICES '08)*, pp. 23-32, 2008.
[2] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer and Tsinghua Univ. Press, July 2007.
[3] *TSC Taxonomy*, IEEE Computer Society, http://www.computer.org/portal/pages/transactions/tsc/mc/tsc_taxonomy.html, 2008.

APPENDIX

Detailed taxonomy structure
M. Services Computing

M. *Services Computing*

0. General
1. Principle of Services
- 0. General
  - a. Services Systems
  - b. Services Models
  - c. Services Technologies
  - d. Services Architectures
  - e. Optimization of Services Systems

2. Services Lifecycle
- 0. General
  - a. Consulting and Strategic Planning
  - b. Services Engagement
  - c. Services Delivery
  - d. Services Operation
  - e. Services Billing
  - f. Services Management
- 1. Key Factors in Services Lifecycle
  - a. Data/Information
  - b. Processes
  - c. People
  - d. Resources
  - e. Financial Factors
  - f. Knowledge and Skills
  - g. Innovation and Technology
- 2. Service-Oriented Business Models
  - a. Services Modernization
  - b. Software as a Service
  - c. Services As Software

3. Web Services
- 0. General
  - a. Web Services Modeling
  - b. Web Services Communication Protocols
  - c. Web Services Binding
  - d. Web Services Publishing
  - e. Stateful Web Services
  - f. Web Services Interoperability
- 1. Composite Services
  - a. Composite Web Services
  - b. Representation of Composite Services
  - c. Multi-Dimensional Modeling
- 2. Web Services Publishing
  - a. Public Services Registry
  - b. Private Services Registry
  - c. Distributed Services Registry
- 3. Web Services Discovery
  - a. Services Discovery Language
  - b. Services Discovery Engine
  - c. Services Discovery Process and Methodology
  - d. Services Discovery Architecture
  - e. Federated Services Discovery

4. Service-Oriented Architecture
- 0. General
  - a. Operational Model
  - b. Realization
- 1. Services Invocation
  - a. Simple Services Invocation
  - b. Metadata of Services Interfaces
  - c. Metadata Publishing
  - d. Advanced Services Invocation Framework
- 2. Bridging Business and IT Architecture
  - a. Enterprise Level Transformation
  - b. Process Level Transformation
  - c. Programming Level Transformation
- 3. Solution Lifecycle
  - a. Solution Modeling
  - b. Solution Development
  - c. Solution Deployment
  - d. Solution Publishing
  - e. Solution Discovery
  - f. Solution Invocation
  - g. Solution Composition
  - h. Collaborations in Solution
  - i. Solution Monitoring
  - j. Solution Management
- 4. Solution Reference Architectures
  - a. Architecture Overview Diagram
  - b. User Interaction and Presentation
  - c. Processes
  - d. Services
  - e. Services Components
  - f. Operational Systems
  - g. Integration
  - h. Quality of Services
  - i. Data Architecture
  - j. Governance

5. Services Relationships
- 0. General
  - a. Relationships in Services Registries
  - b. Relationship Specification Languages
- 1. Web Services Relationship Language
  - a. Relationship Modeling Schema
  - b. Layered Services Relationship Modeling
  - c. Extensions
- 2. Service-Oriented Relationship Modeling
  - a. Business Services Relationship
  - b. Modeling at Business Entity Level
  - c. Modeling at Business Service Level
  - d. Relationship Enriched Services Registry

6. Services Composition
- 0. General
  - a. Aspects of Business Requirements
  - b. Business Requirements Modeling
  - c. Requirements Driven Services Discovery
  - d. Formalization of Services Composition
- 1. Services Integration Framework
  - a. Services Integration Procedure
  - b. Optimization of Services Composition
- 2. Services Value Chain Collaboration
  - a. Inter-Enterprise Collaboration
  - b. Intra-Enterprise Collaboration
  - c. Extended Business Collaboration Model
  - d. Annotated Business HyperChain
  - e. Web Services Collaboration Resources
  - f. Collaborative Message Primitives
  - g. Collaboration Constructs
  - h. Collaborative Exchange Protocols

7. Business Process Management and Integration
- 0. General
  - a. Business Process Modeling
  - b. Business Process Management
- 1. Service-Oriented Business Process Management
  - a. Top-Down Process Management
  - b. Bottom-Up Process Management
  - c. Business Process Reengineering
  - d. Process Reengineering Methodology
- 2. Flexible Business Process Integration
  - a. Lifecycle of an Integration Activity
  - b. Integration Activity Modeling
  - c. Business Process Monitoring

8. Business Grid and Cloud Computing
- 0. General
  - a. Service-Oriented Grid Computing
  - b. Business Grid Solution Framework
  - c. Cloud Computing
- 1. Logical Grid Infrastructure
  - a. Packaged Application Grid
  - b. Business Grid Middleware
  - c. Business Process Grid
- 2. Business Grid Solution Development
  - a. Business Grid Service Development
  - b. Business Grid Service Invocation

9. Enterprise Modeling and Management
- 0. General
  - a. Dynamics of Services Ecosystem
  - b. Requirements for Enterprise Modeling
- 1. Methodologies for Enterprise Modeling
  - a. Balanced Scorecard and Strategy Map
  - b. Component Business Modeling Circle
  - c. Enterprise Architecture
  - d. Enterprise Transformation
- 2. Enterprise Performance Management
  - a. Enterprise Project Management
  - b. Performance Management
  - c. Service-Oriented Enterprise Management
  - d. Enterprise Portfolio Management

10. Service-Oriented Consulting Methodology
- 0. General
  - a. Consulting Method for Strategic Change
  - b. Consulting Method for IT Strategic Plan
- 1. Service-Oriented Business Consulting
  - a. Gap Analysis
  - b. Initiatives Identification
  - c. Value Chain Analysis
  - d. Business Case Analysis
  - e. Portfolio Analysis
  - f. Transition Planning
  - g. Project Management and Collaboration
  - h. IT Service Management

11. Services Delivery Platform and Methodology
- 0. General
  - a. Services Delivery Mechanisms
  - b. Services Engineering
- 1. Service-Oriented Services Delivery Platform
  - a. Traditional Services Delivery Platform
  - b. Collaborative Services Delivery Platform
  - c. Common Services
- 2. Services Delivery Methodology
  - a. Services Delivery Readiness Phase
  - b. Services Delivery Creation Phase
  - c. Services Delivery Operation
- 3. Software as a Service
  - a. Web 2.0 and Web X.0
  - b. Service Mash-up
  - c. New Business Models
- 4. Services as Software
  - a. Asset-based Services Model
  - b. Services Software

12. Application Services and Standards
- 0. General
  - a. Case Studies in Industry
  - b. Case Studies in Scientific Applications
  - c. Case Studies in Government
- 1. Solution-Level Quality of Service
  - a. Context-Aware QoS Model
  - b. Representation of QoS Model
  - c. QoS Data Management
  - d. Business Relationship Model
  - e. Solution-Level QoS Framework
- 2. Data Architecture Framework
  - a. Constructs in Data Architecture
  - b. Relationships Between Constructs
  - c. Data Services
- 3. QoS Management Modeling
  - a. Modeling of Resources
  - b. Modeling the QoS Assurance Process
- 4. Web Services Standard Stack
  - a. Transport
  - b. Messaging
  - c. Description/Publishing/Discovery
  - d. Quality of Service
  - e. Service Composition
- 5. Industry-Specific Standards
  - a. Service-Oriented Solution Reference Architecture
  - b. New Standards
  - c. Case Studies

13. Security, Privacy, and Trust in Services
- 0. General
  - a. Security Concerns of Service-Oriented Solutions
  - b. Privacy Concerns of Service-Oriented Solutions
  - c. Trust in Service-Oriented Solutions
- 1. Access Control in Services Systems
  - a. Role-Based Access Control
  - b. Policy-Based Access Control
  - c. Single-Sign-On
- 2. Security Enablement in Services Systems
  - a. Service-Oriented Security Enablement at Software Level
  - b. Service-Oriented Security Enablement at Hardware Level
  - c. Service-Oriented Security Enablement at Solution Level
  - d. Security Enablement Methods and Tools
- 3. Privacy Management in Services Systems
  - a. Privacy Management in Data Collection
  - b. Privacy Management in Data Transformation
  - c. Privacy Management in Data Dissemination
  - d. Privacy Governance Methods and Tools

14. IT Services Management
- 0. General
  - a. Management of Services Design
  - b. Management of Services Delivery

1. Application Management in Services
   a. Application Management Services
   b. Application Development Services
   c. Legacy Application Transformation in Services
2. Infrastructure Management in Services
   a. Maturity Assessment in Services
   b. Network Management Services
   c. Server Management Services
   d. Data Center Management Services
3. Business and IT Aligned Management Services
   a. Service-Level Agreement for Contracts
   b. Key Performance Indicators for Business Processes
   c. Quality of Services for Services Offerings
   d. Management Methods and Tools for Business and IT Alignment

An Optimal-Control-Based Decision-Making Model and Consulting Methodology for Service Enterprises

Liang-Jie Zhang, *Senior Member, IEEE,* Zhe Shan, *Student Member, IEEE,* and Zhi-Hong Mao, *Member, IEEE*

***Abstract*—The service industry has become a primary point of growth in most countries. However, the current service business is largely constrained by human factors and lacks automated and quantitative techniques for operation and decision-making. In this paper, we propose an optimal-control-based decision-making model that facilitates performance analysis and strategy planning for service enterprises. Using approximate dynamic programming, we are able to handle the model's complexity and obtain a near-optimal solution for the decision-making of employee management, advertisement, and asset investment. We validate our model and approach in a business-simulation practice. Furthermore, we propose a consulting methodology based on the optimal-control model and describe its application in service consulting practices.**

***Index Terms*—Approximate dynamic programming (ADP), consulting methodology, decision-making, optimal control, service enterprises.**

I. Introduction

Services have become the main theme of enormous economic activities in the world. By definition service represents a type of action, performance, or promise that is exchanged for value between provider and consumer [1]. The service industry, or service sector, is often defined as whatever is not agriculture or manufacturing [2]. The service industry encompasses a variety of economic activities in communication, education, finance, healthcare, insurance, logistics, transportation, etc. [3]. Even in the manufacturing industry, service functions are important components for improving customer values and business competitiveness [3]. In fact, the growth of the service industry is partly due to the specialization and outsourcing of service activities inside the manufacturing industry [1]. The service industry has been fast-growing in many countries. In the United States, for example, the service industry accounted for 55% of the total economic activity in 2006 [4] and contributed about 78.7% of the GDP in 2007 [5]. Therefore, promotion of innovation and productivity in the service industry will have significant impact on economy.

Regardless of its important economic role, the service industry is still the least-studied part of the economy [1]. Service science is still an uncharted discipline [3], lacking mature methodology to guide the planning, designing, marketing, and delivering of services to meet the needs of today's economy [1], [6]. Compared with manufacturing systems, service systems are harder to investigate mathematically, because humans in services are more complex to model than physical goods; humans are subject to physiological, psychological, and sociological constraints [6]. Most service businesses today are still labor-intensive and rely on skilled human workers to deliver services to consumers. Investment decisions and service deliveries are heavily limited by personal knowledge, expertise, and biases. In order to improve productivity of service business, analytical and quantitative methods that enable service enterprises to deliver cost-effective services are highly desirable.

Services computing has been proposed as a multidisciplinary area that explores the application of computing and information technology in the design, operation, and management of business services [7]. This area attracts tremendous interest and effort in both scientific investigation and industrial application. However, the mathematical foundation of service computing has not received sufficient attention. Specifically, there is a substantial need for formalized ways to represent service enterprises and facilitate decision-making in service enterprises. Business architecture has been introduced to break an enterprise down into components based on business functionalities and time-varying requirements from the market [8], [9]. However, most work in this direction provides only text-rich descriptions and suggestions. It is still a challenging issue to formalize quantitatively a way of guiding effective decision-making in service enterprises.

In this paper, we aim to formalize a unified decision-making model for service enterprises based on the optimal control theory. We apply optimal control to investigate specific types of service enterprises that are project-based and manpower-centric business systems, but our method can be generalized to other types of service enterprises. We also propose an effective algorithm to solve the optimal decision-making problems in service enterprises. The contribution of this paper is significant, because it takes a first step toward (i) building mathematical foundation for services computing and (ii) providing a quantitative framework to solve complex problems of inter-disciplinary nature in service science. Specifically, our study and future work along this direction address the need for mathematical models of service enterprises and systematic tools that facilitate human decision-making in the control and management of service enterprises.

Our work is built upon the current advances in enter-

Manuscript received Octobor 17, 2008; revised May 10, August 5, and September 5, 2009. This work was supported by the National Science Foundation (NSF) under Grant CMMI-0727256.

L.-J. Zhang is with the IBM T. J. Watson Research Center, Hawthorne, NY 10532 USA (zhanglj@us.ibm.com).

Z. Shan is with the Department of Supply Chain and Information Systems, Pennsylvania State University, University Park, PA 16802 USA (zheshan@psu.edu).

Z.-H. Mao is with the Department of Electrical and Computer Engineering and the Department of Bioengineering, University of Pittsburgh, Pittsburgh, PA 15261 USA (e-mail: maozh@engr.pitt.edu).

prise modeling and the application of optimal control in engineering management. A variety of models has already been developed for a variety of enterprises other than service enterprises (e.g. [10]–[12]). These models are enlightening to this study. In general an enterprise model is an abstract representation of the structure, information, processes, and resources of an enterprise [13]. An enterprise is a unit of economic organization that develops and delivers products or services to customers. Since an enterprise usually has complex organization and operation, there have been a vast number of approaches for enterprise modeling at different levels [14]. For example, at the function level, enterprise modeling often seeks to build a graphical representation of the function of an enterprise [15]. At the data level, the modeling aims to create a data model implemented in a database and define business requirements for that database [16]. At the level of business or manufacturing processes, enterprise modeling is to build representation (e.g. graphs or discrete events) of the processes of an enterprise, so that the current processes can be analyzed and improved in their capabilities and efficiency [11]. At the level of human resources, the modeling aims to characterize and analyze the states (e.g. number, skill-level, and competence) of the people in an enterprise [11], [12]. These models at different levels promote understanding of an enterprise. Especially, when an enterprise is represented by a quantitative or mathematical model, optimization methods can be naturally developed to improve the productivity, quality, and efficiency of the enterprise.

Optimal control is a special optimization method for control of dynamic systems, i.e., systems that evolve over time. This technique has a wide range of applications in economics and engineering management [17]. Dorfman illustrated in [17] that optimal control is formally identical to capital theory, and control-theoretic frameworks can be developed to represent and analyze the generic economic ecosystems. Wu et al. [18] proposed an effective strategy utilizing stylized models and optimal control for the development of modular products. They considered a reuse-redesign decision at the product-component level and applied optimal control to figure out the optimal solution for minimizing the overall development cost for each component. Mendez and Narasimhan [19] examined the market-oriented aspects of the cost of quality and proposed an optimal-control-based method pursuing joint examination of the direct and indirect effects on unit cost. Tu et al. [20] investigated a manufacturing paradigm that aims to achieve economies of both customization (scope) and mass production (scale). They developed a cost-index structure for production cost in mass customization and used optimal control to determine the best selection of alternative operation routines and suppliers. Terwiesch and Xu [21] addressed the issues encountered during a production ramp-up period and formalized the tradeoff between learning and process change in the form of an optimal-control or dynamic-optimization problem. Enlightened by the above examples, we aim to develop optimal control models in order to find optimal solutions for sequential decision-making in service enterprises.

The rest of this paper is organized as follows: Section II presents an optimal-control model for decision-making in service enterprises. The model captures major behaviors of a service enterprise. Section III utilizes dynamic programming to solve the optimal control problem and develops a variant of the forward-induction algorithm using cubic spline interpolation to construct value functions. Section IV illustrates the capability of the optimal-control model via a case study. Section V introduces a consulting methodology that helps decision-makers or consultants to make investment and planning decisions based on the optimal-control-based decision-making model. Section VI presents discussions on several related issues and ongoing efforts. The last section concludes the paper.

II. Decision-Making Model for Service Enterprises (DMM4SE)

In this section, we model the decision-making of a service enterprise based on optimal control. Our objective is to help business managers or consultants to quantify their available options and make decisions in an effective and managed fashion.

Over a certain period of time, say from period 1 to K, in a service enterprise, the decision-makers need to choose a sequence of actions so as to maximize some "reward" or optimize the enterprise performance with respect to some predetermined criterion. Without loss of generality, the time interval is set to a year in this paper, but it can be chosen based on specific management requirement, e.g., a month or a quarter. At time $k \in [1, K]$, the enterprise inherits a certain asset and other conditions from its previous period. These can be described by a set of variables, and these variables correspond to a system's *state variables*, or simply *states*. Denote the state variables at time k by a vector $x(k)$. Correspondingly, the initial state is denoted as $x(0)$ or x_0, and the terminal state is $x(K)$. With this state x and at that particular time k, the enterprise takes some decisions concerning investment for advertisement and policies for hiring, training, and laying off employees, etc. Denote the decisions taken at any time k by a vector $u(k)$. This decision vector u corresponds to the input to the dynamic system modeling the operation of the enterprise. From the state at a specified time together with the specified current decisions, the enterprise derives a certain reward:

$$r(x(k), u(k)).$$

This reward determines the benefit earned at time k as a result of having the state of $x(k)$ and taking decisions $u(k)$.

Note that the decisions taken at any time influence not only the rewards earned at that time but also the system state. The current state of a service enterprise can be expressed as a function of the previous state and current decisions:

$$x(k) = f(x(k-1), u(k)). \tag{1}$$

Given the initial state x_0, the total rewards that will be earned from the initial time ($k = 1$) to the terminal time K is given by

$$V_K(x_0, \mathbf{u}) = \sum_{k=1}^{K} r(x(k), u(k)) \tag{2}$$

where $\mathbf{u}$ denote the entire time path of the decision variable u from the initial time to K.

The above formulas express the essence of the problem of making decisions in a dynamic context. The problem is to select the time path $\mathbf{u}$ so as to make the total reward V_K as large as possible. This problem is exactly an optimal control problem. In the following subsections, we will introduce the details of this model.

A. Key Components of DMM4SE

Most of the enterprises today are project-based businesses. The projects can be internal projects on basic research, product development, communication, IT infrastructure transformation, and business process reengineering. The projects can also be external projects including, for example, IT outsourcing and on-site service delivery. Most of the revenues of service providers come from service projects. Without loss of generality, we assume that the annual gross income of a service enterprise under consideration is contributed from the projects (or contracts) that the enterprise receives within a year.

To describe the dynamics of the operation and decision-making in a service enterprise, we introduce the following notation:

- *Indices:*
  - k: year number;
  - j: year of experience of an employee.
- *Parameters:*
  - K: maximum decision period;
  - $n_{\max}$: maximum years of experience beyond which an employee does not stay;
  - $w_{\mathrm{H}}(j)$: cost of hiring an employee with j years of experience;
  - $w_{\mathrm{L}}(j)$: layoff cost for an employee with j years of experience;
  - w_{R}: allowance paid to a retiring employee;
  - $w_{\mathrm{S}}(j)$: annual salary paid to an employee with j years of experience;
  - $\delta_{\mathrm{S}}(c, j)$: average incremental strength of an employee with j years of experience after receiving a training with cost c per person.
- *Decision (or input) variables:*
  - $u_{\mathrm{A}}(k)$: advertising expenses in year k;
  - $u_{\mathrm{LE}}(k, j)$: in year k, the number of employees that are laid off with j years of experiences ($1 \le j \le n_{\max} - 1$);
  - $u_{\mathrm{NE}}(k, j)$: in year k, the number of new employees that are hired with j years of experience ($0 \le j \le n_{\max} - 1$);
  - $u_{\mathrm{P}}(k)$: project price offered by the enterprise in year k;
  - $u_{\mathrm{T}}(k, j)$: in year k, the expense that the enterprise spends in training employees with j years of experience ($0 \le j \le n_{\max} - 1$);
  - $u_{\mathrm{Z}}(k)$: asset investment in year k.
- *State variables:*
  - $x_{\mathrm{A}}(k)$: advertising capital in year k, which summarizes the effects of current and past advertising investment;
  - $x_{\mathrm{E}}(k, j)$: in year k, the number of employees that are kept with j years of experience ($0 \le j \le n_{\max}$);
  - $x_{\mathrm{S}}(k, j)$: average strength of an employee with j years of experience in year k ($0 \le j \le n_{\max}$);
  - $x_{\mathrm{Z}}(k)$: asset of the enterprise in the year k.
- *Output variables:*
  - $y_{\mathrm{O}}(k)$: operational cost in year k;
  - $y_{\mathrm{P}}(k)$: number of projects received by the enterprise in year k.

Furthermore, the state vector $x(k)$ and decision vector $u(k)$ are defined as

$$\begin{aligned} x(k) &= (x_{\mathrm{A}}(k),\ x_{\mathrm{E}}(k,0), ..., x_{\mathrm{E}}(k, n_{\max}), \\ &\quad x_{\mathrm{S}}(k,0), ..., x_{\mathrm{S}}(k, n_{\max}), x_{\mathrm{Z}}(k))' \\ u(k) &= (u_{\mathrm{A}}(k), u_{\mathrm{LE}}(k,0), ..., u_{\mathrm{LE}}(k, n_{\max}-1), \\ &\quad u_{\mathrm{NE}}(k,1), ..., u_{\mathrm{NE}}(k, n_{\max}-1), \\ &\quad u_{\mathrm{P}}(k), u_{\mathrm{T}}(k,0), ..., u_{\mathrm{T}}(k, n_{\max}-1), u_{\mathrm{Z}}(k))' \end{aligned}$$

where $'$ represents transpose.

In the following subsections, we will introduce the submodels characterizing the dynamics of the state variables.

B. Advertising Capital

We follow the model by Nerlove and Arrow [22] to describe the dynamics of advertising capital (or called market goodwill). We assume that the advertising capital depreciates over time at a constant proportional rate δ_{A}. Then we have

$$x_{\mathrm{A}}(k) = (1 - \delta_{\mathrm{A}}) x_{\mathrm{A}}(k-1) + u_{\mathrm{A}}(k). \tag{3}$$

The above equation can also be written as

$$x_{\mathrm{A}}(k) - x_{\mathrm{A}}(k-1) = u_{\mathrm{A}}(k) - \delta_{\mathrm{A}} x_{\mathrm{A}}(k-1)$$

which implies that the net investment in advertising is the difference between gross investment $u_{\mathrm{A}}(k)$ and depreciation [23].

C. Manpower Dynamics

Inspired by Aksin's work on optimal policies for improving workers' productivity [24], we develop a set of equations for managing manpower dynamics. Let us first consider the update of employee numbers:

$$x_{\mathrm{E}}(k, j) = \begin{cases} u_{\mathrm{NE}}(k, 0) \ \text{ if } j = 0 \\ x_{\mathrm{E}}(k-1, j-1) + u_{\mathrm{NE}}(k, j) - u_{\mathrm{LE}}(k, j) \\ \qquad\qquad \text{if } 1 \le j \le n_{\max} - 1 \\ x_{\mathrm{E}}(k-1, j-1) \ \text{ if } j = n_{\max}. \end{cases} \tag{4}$$

The above equation can be interpreted as follows. For the case of $1 \le j \le n_{\max} - 1$, $x_{\mathrm{E}}(k, j)$, the number of employees that are kept with j years of experience in the k-th year, should include (i) the number of employees that had $j - 1$ years of experience in last year (year $k - 1$) and (ii) the number of new employees that are hired with j years of experience in this year, but should exclude the number of employees that are laid off this year with j years of experience. For the case of $j = 0$, $x_{\mathrm{E}}(k, 0)$ simply means the number of employees without any work experience in year k, and it should equal the number of new employees that are hired this year without

any previous work experience. Finally, for $j = n_{\max}$, it is unlikely that an enterprise recruits or lays off someone who will retire immediately in next year, so $x_{\mathrm{E}}(k, n_{\max})$ includes just one component, i.e., the number of employees that had $n_{\max} - 1$ years of experience in the past year.

Next we consider the update for the average strength or productivity of an employee:

$$x_{\mathrm{S}}(k,j) = \begin{cases} \delta_{\mathrm{S}}\left(\frac{u_{\mathrm{T}}(k,0)}{u_{\mathrm{NE}}(k,0)}, 0\right) & \text{if } j = 0 \\ x_{\mathrm{S}}(k-1,j-1) + \delta_{\mathrm{S}}\left(\frac{u_{\mathrm{T}}(k,j)}{x_{\mathrm{E}}(k,j)}, j\right) & \\ \qquad\qquad \text{if } 1 \le j \le n_{\max} - 1 & \\ x_{\mathrm{S}}(k-1,j-1) + \delta_{\mathrm{S}}\left(0, n_{\max}\right) & \\ \qquad\qquad \text{if } j = n_{\max}. & \end{cases} \tag{5}$$

For the most common case where $1 \le j \le n_{\max}-1$, $x_{\mathrm{S}}(k,j)$, the average strength of an employee with j years of experience in the k-th year, should inherit the strength from last year, i.e., $x_{\mathrm{S}}(k-1,j-1)$, and at the same time add on it the incremental strength after receiving a training with cost $u_{\mathrm{T}}(k,j)$. This incremental part is denoted $\delta_{\mathrm{S}}\left(\frac{u_{\mathrm{T}}(k,j)}{x_{\mathrm{E}}(k,j)}, j\right)$, whose form is to be determined via investigation of some empirical data. For the case of $j = 0$, $x_{\mathrm{S}}(k,0)$ cannot inherit anything from the past and has to purely rely on the incremental strength obtained from training. For $j = n_{\max}$, it is unlikely that an enterprise would train someone who will retire immediately in next year, so $u_{\mathrm{T}}(k, n_{\max})$ is zero. Note that $\delta_{\mathrm{S}}\left(\frac{u_{\mathrm{T}}(k,j)}{x_{\mathrm{E}}(k,j)}, j\right)$ can be greater than zero even if the training cost $u_{\mathrm{T}}(k,j)$ is zero, because the strength of an employee can increase with gain of work experience even without specialized training. Moreover, we assume that over time there is no forgetting in employee's skill strength acquired by training.

D. Assets

The service assets include both hardware and software facilities supporting service operations, such as IT servers, storages, and information databases. Business process is another type of assets. It formalizes a set of service operations and coordinates human and IT resources. A good business process can help optimize service operations. These service assets are complementary to the capability of human employees. In this paper, we assume that the assets depreciate over time at a constant rate δ_{Z} but will increase following additional investment $u_{\mathrm{Z}}(k)$:

$$x_{\mathrm{Z}}(k) = (1 - \delta_{\mathrm{Z}}) x_{\mathrm{Z}}(k-1) + u_{\mathrm{Z}}(k). \tag{6}$$

E. Service Projects

As mentioned previously, the service enterprises considered here are project-based business. To win projects, a service enterprise has to compete with its peers in the market. The market position of the enterprise determines how many customers in the market will purchase its services. We define the amount of service orders from customers as *project demand*. Sometimes not all the orders can be handled by the service enterprise, because the capability of the enterprise is limited by its assets as well as the number and skill-levels of its employees. We define this capability of an enterprise in service as *project capacity*. If the service project demand is higher than service project capacity, the superfluous service orders will be lost. Consequently, the customer satisfaction-level will drop, and the advertising capital will decrease. If the project demand is lower than service project capacity, some employees have to be idle, and the enterprise wastes its human resources. Therefore, a service enterprise needs to match the dynamic demand with its service capacity.

We assume that all projects need to go through a bidding process. According to [23], the rate of project demand depends on the market position of the advertising capital, project price, and other variables not under the control of the service enterprise, such as consumer incomes, population, etc. Following [22], we use a simplified model for project demand:

$$D(k) = n_{\mathrm{P}}(k) d(k) \tag{7}$$

where $D(k)$ represents the enterprise's project demand in year k, $n_{\mathrm{P}}(k)$ denotes the total number of projects available in year k, and $d(k)$ is determined by

$$d(k) = a \left[\frac{u_{\mathrm{P}}(k)}{U_{\mathrm{P}}(k)}\right]^{-\eta} \left[\frac{x_{\mathrm{A}}(k)}{X_{\mathrm{A}}(k)}\right]^{\beta} \tag{8}$$

where η and β are defined as the elasticities of project demand with respect to price and advertising capital, respectively; $u_{\mathrm{P}}(k)$ is the project price offered by the enterprise in year k, and $U_{\mathrm{P}}(k)$ the sum of project prices offered by all the enterprises in the corresponding market; $x_{\mathrm{A}}(k)$ denotes the advertising capital in year k, and $X_{\mathrm{A}}(k)$ the total advertising capital of the whole market; and a is a normalization parameter such that the sum of $d(k)$ of all the service enterprises in the same market equals 1. This model has been empirically observed in many industries [22].

The project capacity of a service enterprise in year k, denoted $C(k)$, relies on the number of employee, their experiences and strengths, and the assets of the enterprise:

$$\begin{aligned} C(k) \quad = \quad & C\left(\left[x_{\mathrm{E}}(k,0), ..., x_{\mathrm{E}}(k,n_{\max})\right], \right. \\ & \left. \left[x_{\mathrm{S}}(k,0), ..., x_{\mathrm{S}}(k,n_{\max})\right], x_{\mathrm{Z}}(k)\right) \end{aligned} \tag{9}$$

where the function $C(\,\cdot\,)$ needs to be determined from the empirical data.

The number of projects received by the enterprise in year k, i.e., y_{P}, should be limited by both the project demand $D(k)$ and project capacity $C(k)$:

$$y_{\mathrm{P}}(k) = \min\{D(k), C(k)\}. \tag{10}$$

If $D(k)$ is larger than $C(k)$, the service enterprise does not have enough resources to support all the demands. For simplicity, we assume in the model that all the back orders will be lost. If $C(k)$ is larger, the enterprise is able to handle all the project demands, and thus $y_{\mathrm{P}}(k)$ equals $D(k)$. In the latter case, the extra resources will be idle, but the enterprise still needs to pay the expenses of these extra capacity, e.g., the salary of idle employee and the maintenance fee of hardware.

F. Income, Cost, and Reward

The total profit made by the enterprise in year k should equal the revenue gained from the projects received by the enterprise subtracted by

- Human resource expenses, which include (i) salaries paid to employee, (ii) training cost, (iii) hiring cost, (iv) layoff cost, and (v) retirement cost.
- Marketing spending, which includes advertising expense.
- Operational cost, which includes the office administration fee, project supporting expenses, and computing infrastructure cost. More people or more projects the enterprise has, more money it needs to pay for service operations. Meanwhile, the advanced assets such as SOA (service-oriented architecture)-based enterprise architecture and state-of-the-art service delivery processes can reduce the operational cost. Therefore, the operational cost is a function of the employee number, project number, and assets:

$$y_{\mathrm{O}}(k) = f_{\mathrm{O}}(x_{\mathrm{E}}(k), y_{\mathrm{P}}(k), x_{\mathrm{Z}}(k)) \tag{11}$$

  where $f_{\mathrm{O}}(\cdot)$ need to be determined from the empirical study.

Therefore, the profit (reward) of a service enterprise in year k can be represented as

$$\begin{aligned} r(x(k), u(k)) = {} & u_{\mathrm{P}}(k) y_{\mathrm{P}}(k) - u_{\mathrm{A}}(k) - y_{\mathrm{O}}(k) \\ & - \sum_{j=0}^{n_{\max}} w_{\mathrm{S}}(j) x_{\mathrm{E}}(k,j) - \sum_{j=0}^{n_{\max}-1} u_{\mathrm{T}}(k,j) \\ & - \sum_{j=0}^{n_{\max}-1} w_{\mathrm{H}}(j) u_{\mathrm{NE}}(k,j) - \sum_{j=1}^{n_{\max}-1} w_{\mathrm{L}}(j) u_{\mathrm{LE}}(k,j) \\ & - w_{\mathrm{R}} x_{\mathrm{E}}(k, n_{\max}). \end{aligned} \tag{12}$$

G. Optimality Criteria and Service Enterprise Index

As a result of choosing and implementing a policy, the service enterprise receives rewards in year $1, ..., K$. The business objective is to choose a series of decisions in order to maximize the accumulated rewards. In the service industry, the performance measures consist of multiple assessment elements, e.g., efficiency, productivity, and enterprise-wide profitability. We need to build a multi-factor quantitative performance evaluation framework to monitor the performance of service enterprise.

Multiple indicators of performance evaluation obviously lead to conflicts. Short-term and long-term trade-offs need to be considered. For example, by reducing marketing and training expenses, current costs will decrease and profits will increase. However, these might be exactly the wrong things to do to maximize long-term profitability. Considering the short-term and long-term trade-offs, we propose to use *service enterprise index* (SEI) to reflect a specific business goal. For this paper and specifically the case study in Section IV, we consider a simplified SEI, which is the total profits in a given decision period. Our objective is to

$$\text{maximize} \quad V_K(x_0, \mathbf{u}) = \sum_{k=1}^{K} r(x(k), u(k)) \tag{13}$$

subject to (3)-(12).

III. Dynamic Programming Formulation and Algorithm for DMM4SE

In this section, we formulate the above optimal-control problem into a dynamic programming (DP) problem. However, the computation required for finding the optimal solutions increases exponentially with the number of state variables. Therefore, we develop an approximate dynamic programming (ADP) algorithm which uses cubic-spline interpolation to simplify the computation.

A. DP Formulation

The firm starts the business period with an initial state $x(0) = x_0$. Let policy $\mathbf{u}$ denote the series of decisions at all periods (from 1 to K). Denote $\mathbf{\Pi}$ the space of the policies, and define the optimal value function of the total profits as

$$V_K^*(x_0) = \max_{\mathbf{u} \in \mathbf{\Pi}} V_K(x_0, \mathbf{u}). \tag{14}$$

Policy $\mathbf{u}^*(x_0)$ is said to be the optimal policy (given the initial state x_0) if

$$V_K(x_0, \mathbf{u}^*(x_0)) = V_K^*(x_0). \tag{15}$$

We use forward induction to solve the optimal control problem, since our problem has a fixed initial state x_0 and a floating terminal state. For any $k \in \{1, ..., K\}$ and given state $x(k)$, let $U_k(x(k))$ denote the optimal profit accumulated from year 1 to year k while the state variable evolves from x_0 to $x(k)$. Then we have the following optimality condition for $U_k(x(k))$:

$$U_k(x(k)) = \max_{\substack{x(k-1),\, u(k) \text{ subject to} \\ f(x(k-1), u(k)) = x(k)}} \{r(x(k), u(k)) + U_{k-1}(x(k-1))\} \tag{16}$$

where $f(\cdot)$ is defined in (1). In year 0, the enterprise receives no profit, so the boundary condition is $U_0(x_0) = 0$. Based on the above definitions, $V_K^*(x_0)$ equals the maximum value of $U_K(x(K))$ over all possible final state $x(K)$ evolved from x_0.

B. ADP Algorithm

Solving the above DP problem is computationally intensive for high dimensional state space. Therefore, we develop a heuristic method based on ADP (see [25] for a comprehensive review of ADP). We utilize an interpolation-based algorithm to reduce the time and memory required of computation. The standard forward induction method [26] stores the value function for all possible states of a DP in each period; our method stores the optimal values for only a small subset of the entire state space (which we refer to as anchor values) and approximates the values of other states through *ad hoc* spline interpolation. The spline interpolation method produces interpolants that are simple, yet flexible and smooth piecewise polynomial functions [27]. In particular, we use cubic splines to interpolate the multidimensional value function of the DMM4SE. Our algorithm embeds these interpolated values within a standard forward induction method and generates

near-optimal solutions. Similar algorithms have been successfully implemented to reduce computational complexity in large-dimensional dynamic programs [28], [29].

Let us introduce some notation. Define Γ as the state space, which contains all possible values of the state variable x, and let n be the total number of the dimensions of Γ. Denote γ_i the number of anchor points for the cubic spline interpolation in the i-th dimension of the state space, and denote $\hat{\Gamma}_i$ the set of these anchor points in the i-th dimension. Define $\hat{\Gamma}$ as $\hat{\Gamma}_1 \times \hat{\Gamma}_2 \times \dots \hat{\Gamma}_n$. This set is a subset of Γ and contains all the points whose coordinates are some anchor points in $\hat{\Gamma}_i$, $i = 1, ..., n$.

The basic idea of our ADP algorithm is as follows. We compute the approximate values of $U_k(x)$, denoted by $\hat{U}_k(x)$, for those anchor points in $\hat{\Gamma}$, using the estimates of $\hat{U}_{k-1}(x)$ and the optimality equation (16). Then we find a multidimensional cubic spline function passing through $\hat{U}_k(x)$ for all $x \in \hat{\Gamma}$. The value of $U_k(x)$ for $x \notin \hat{\Gamma}$ can be approximated by interpolation using the cubic spline function. Initially, we can exactly solve the optimal $U_1(x)$ and let $\hat{U}_1(x) = U_1(x)$. Iteratively, we get the near-optimal values $\hat{U}_k(x)$ for $k = 2$ through K. Then we use backward induction to compute the optimal decision series, i.e., $u^*(K)$, $u^*(K-1)$, ..., $u^*(1)$. Details of the interpolation-based ADP algorithm are described in the following five steps:

1) For $k = 1$, compute $U_1(x)$ based on the optimality equation (16) with boundary condition $U_0(x_0) = 0$, and then let $\hat{U}_1(x) = U_1(x)$.
2) For $k = 2$ to K:
   a) Determine the anchor points for interpolation:
      i) Determine the minimum (denoted $x_{i,\min}$) and maximum (denoted $x_{i,\max}$) for all possible values of x_i, where x_i denotes the i-th element of x.
      ii) Choose γ_i points from $[x_{i,\min}, x_{i,\max}]$ such that these points divide the interval into approximately equal partitions.
   b) For each $x \in \hat{\Gamma}$, compute $\hat{U}_k(x)$ using the optimality equation (16), i.e.,
   $$\hat{U}_k(x) = \max_{\substack{x',\, u \text{ subject to} \\ f(x',u)=x}} \left\{ r(x,u) + \hat{U}_{k-1}(x') \right\}.$$
   c) For $x \notin \hat{\Gamma}$, calculate the approximate value of $\hat{U}_k(x)$:
      i) Construct a cubic spline function $S(x)$ such that $S(x') = \hat{U}_k(x')$ for all $x' \in \hat{\Gamma}$.
      ii) For all $x \in \Gamma \backslash \hat{\Gamma}$, interpolate $\hat{U}_k(x) = S(x)$.
3) For $k = K$, compute $x^*(K) = \arg\max_{x \in \Gamma} \{\hat{U}_K(x)\}$.
4) For $k = K$ to 2:
   a) Construct the cubic spline function $S(x)$ such that $S(x') = \hat{U}_{k-1}(x')$ for all $x' \in \hat{\Gamma}$.
   b) For all $x \in \Gamma \backslash \hat{\Gamma}$, interpolate $\hat{U}_{k-1}(x) = S(x)$.
   c) Compute $x^*(k-1)$ and $u^*(k)$:
   $$\{x^*(k-1), u^*(k)\} = \arg\max_{\substack{x,\, u \text{ subject to} \\ f(x,u)=x^*(k)}} \left\{ r(x^*(k), u) + \hat{U}_{k-1}(x) \right\}.$$
5) For $k = 1$, compute $u^*(1)$ such that $f(x_0, u^*(1)) = x^*(1)$.

IV. Case Study

To illustrate the usage of our optimal-control-based decision-making model, we analyzed a sample dataset from the Beacon business simulation practice [30]. This simulation practice was conducted in the IBM Research's Micro-MBA Program in 2006 and involved four enterprises run by four teams of participants. The four teams competed against each other in the market for various products and services. In the simulation practice, each enterprise had both manufacturing business and service business, but for this study we focused on the service business. In each year, the teams made decisions on the project pricing, marketing spending, employee hiring or laying-off, and employee training. Each team aimed to maximize their total profit within a fixed period. The participants in this simulation practice were all professionals and experts in business operations and service management. Therefore, the dataset can reflect the behavior and dynamics of service enterprises in the real world.

Fig. 1 shows the performance of all the four teams or service enterprises from year 8 to year 12. One of the enterprises, the South Service Inc., or the South for short, outperformed the other three enterprises in terms of profits. Although the project sales [Fig. 1(b)] of the South was not the highest, this team successfully leveraged the employee power [Fig. 1(c)] and project pricing [Fig. 1(d)]. With a relatively higher project price, it maintained the service capacity with a stable employee team. In doing so, it beat other teams in the simulation practice. Therefore, in this case study we chose the South as the target of our analysis so as to demonstrate the effectiveness of our approach: We calculated the optimal policy for this enterprise based on our decision-making model and ADP algorithm and then compared our solution with the recorded performance in the simulation practice.

A. Data Analysis and Model Fitting

The dataset provides us the information about operations of the South from year 5 to year 12. The data include information about the employees, their starting and ending skill-levels, history of their status (whether working or idle), number of employee per contract, and cost per contract. The data also provide information of project sales, price, total revenue, cost of working and idle employees, gross profit, administration cost, expenses in advertising and training, severance payment, operating profit, and lost orders. Furthermore, the data include all the decisions made by the South about project pricing, marketing spending, and employee hiring, firing, and training.

In this case study, the employee's annual salary (w_{S}) was $\$100K$, no matter working or idling. The employee layoff cost (w_{L}) was $\$30K$. There was no cost for hiring new employees. The Beacon dataset does not have the detailed information about the experiences and skill-levels of individual employees. However, the average skill-level of all employees is available for each year.

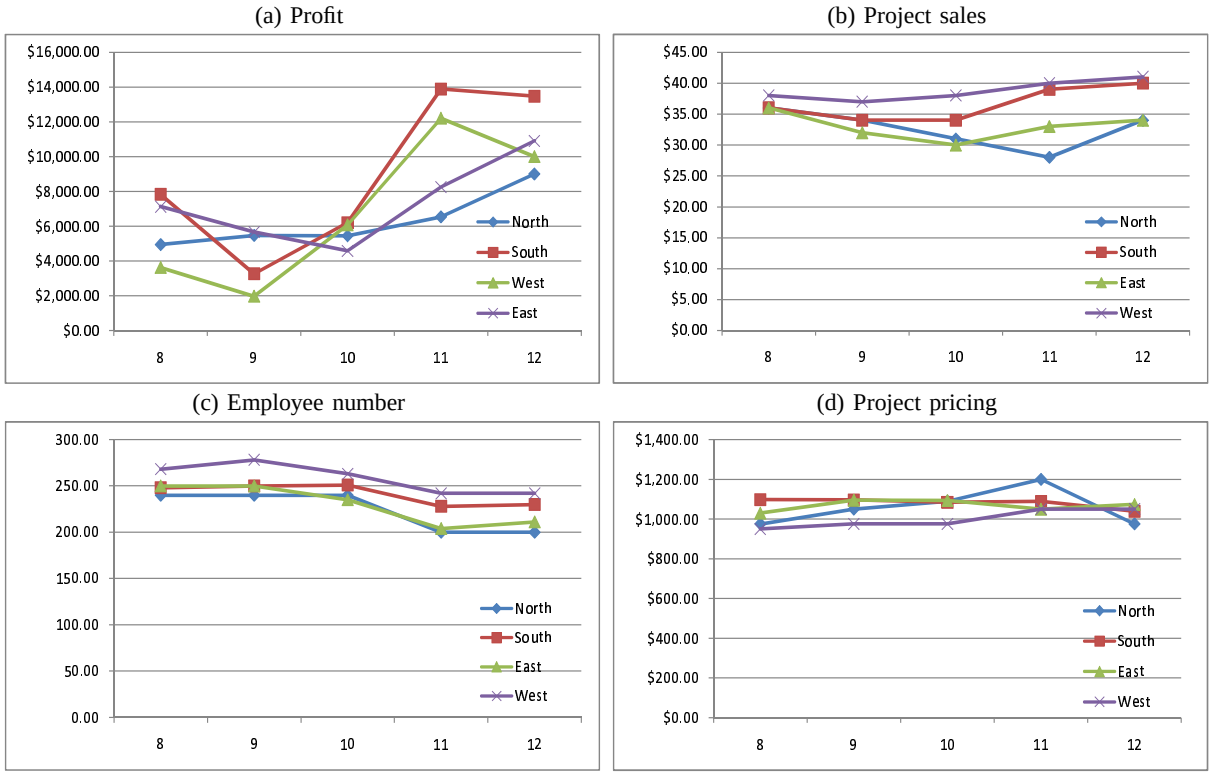


Fig. 1. Performance comparison of four service enterprises

Fig. 2. Data analysis of the Beacon project.

Fig. 2(a) shows the relation curve for employee skill-level improvement with respect to training expense. We ran a regression and obtained the following function for the skill-level improvement of employees:

$$\delta_{\mathrm{S}}(\frac{u_{\mathrm{T}}(k)}{x_{\mathrm{E}}(k)}) = \frac{-3.81}{\frac{u_{\mathrm{T}}(k)}{x_{\mathrm{E}}(k)} + 21.6} + 0.226$$

where $u_{\mathrm{T}}(k)$ is the total training expense in year k, $x_{\mathrm{E}}(k)$ is the total number of employees in year k, and $\delta_{\mathrm{S}}(\cdot)$ is the average incremental strength of an employee after receiving training of expense $u_{\mathrm{T}}(k)/x_{\mathrm{E}}(k)$ per employee. The above equation was used in the update for the average strength of an employee [see (5)].

Fig. 2(b) shows that the employee number per contract decreased linearly with the increase of the employee skill-level. Therefore, the project capacity of the South can be determined by the number and skill-levels of its employees. Using a linear regression, we had

$$C(k) = \frac{x_{\mathrm{E}}(k)}{-5.21x_{\mathrm{S}}(k) + 14.1}.$$

Fig. 2(c) shows how the project demand varied with the price and advertising spending. Here we use the term "share" to represent the portion of a quantity in the market. For example, price share means the project price offered by the South divided by the sum of project prices offered by all the four enterprises in the simulation practice. We used the curves in Fig. 2(c) to fit (3) and (8) and obtained

$$x_{\mathrm{A}}(k) = 0.99\ x_{\mathrm{A}}(k-1) + u_{\mathrm{A}}(k),$$

and

$$D(k) = 0.22 \left[\frac{u_{\mathrm{P}}(k)}{U_{\mathrm{P}}(k)}\right]^{-0.43} \left[\frac{x_{\mathrm{A}}(k)}{X_{\mathrm{A}}(k)}\right]^{0.33}.$$

Knowing the project capacity, demand, and sales [i.e., $y_{\mathrm{P}}(k)$, the number of projects received by the enterprise], we plot them in Fig. 2(d). The figure shows that project sales were bounded by both the project capacity and demand [see (10)]. In year 5, 7, 9, 10 and 12, the sales were constrained by the project demand. In year 6, 8 and 11, the project sales were limited by the project capacity of the enterprise; in other words, the South did not have enough resources to support all the potential projects in those years. This observation supports our model built in (10).

B. Performance Analysis

Based on the model obtained in Section IV-A, we implemented our interpolation-based ADP algorithm (in MATLAB [31]) to find the optimal policy for the South. In this case study, we had five decision variables: project price, number of hiring employees, number of layoff employees, advertising expense, and training spending; and we had three state variables: employee number, employee average skill-level, and advertising capital. A simplified SEI used here was the total profit over a period of five years.

Fig. 3 presents the human-resource related decisions and consequences produced by our algorithm. For comparison, the figure also plots the original decisions and performance of the South. The employee number was significantly larger in our solution than in the simulation practice of the South [Fig. 3(a)]. Except for year 11, the decision suggested by our algorithm was to hire new employees in each year. The training expense in the our solution experienced a big jump in year 6, but remained at low levels afterwards [Fig. 3(b)]. As a result, the average employee skill-level was improved steadily [Fig. 3(c)], and the project capacity was expanded to a very high level [Fig. 3(d)].

Fig. 4 presents the marketing related decisions and consequences produced by our algorithm. A straightforward way of gaining profit is to increase the project demand. By adjusting the price and increasing the marketing spending, the project demand can be increased. For marketing spending, the decisions suggested by our algorithm had little difference with the original decisions of the South in the simulation practice [Fig. 4(a)]. However, for decisions on pricing, the difference was significant [Fig. 4(b)]. The strategy suggested by our algorithm was to seize large market share by offering low prices. In doing so, the project demand was increased dramatically [Fig. 4(c)]. Note that our solution was derived based on the simplified price-marketing-demand model in a closed market. The real business situation may be much more complicated.

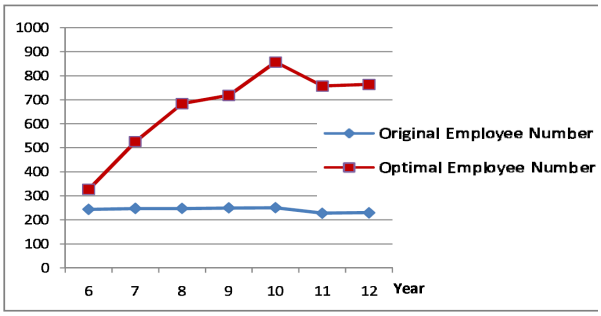
Compared with the results shown in Fig. 2(d), our solution offered a better match between the project capacity and project demand, as demonstrated in Fig. 4(d). This implies that our solution made better use of the enterprise resources (with less wasting or idling of human resources). As a result, there was a big improvement in project sales, which almost tripled the original sales.

Fig. 5 shows the predicted profit in our solution and the original profit of the South in the simulation practice. Except for year 6, the profit produced by our algorithm was much larger than the original profit. In year 6, our solution suggested the enterprise put more efforts in building a strong workforce and augmenting the market share—these efforts established a solid ground for the development of the enterprise in the following years. In summary, the total predicted profit in our solution was $\$106,792K$, which doubled the original profit, $\$53,208K$. This case study has demonstrated the effectiveness of our decision-making model and solution.

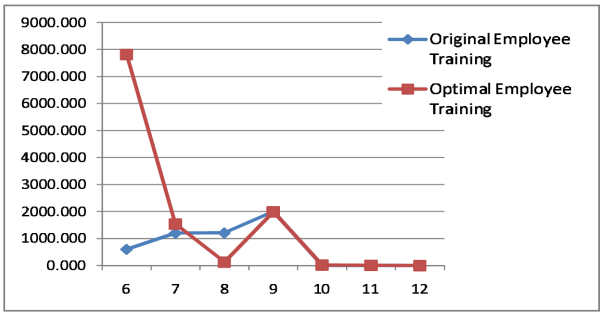
C. Effectiveness of ADP Algorithm

To assess the effectiveness of our ADP algorithm, we compared its performance with that of the standard forward-induction-based DP algorithm. The DP algorithm may find the optimal solutions to the optimal control problems formulated in this paper, but is highly computationally demanding. In the previous case study, our interpolation-based ADP algorithm yielded an outstanding performance: It obtained a near-optimal solution with a gap of less than 0.5% to the actual optimal solution, while using less than 1% of the computer memory and 15% of the CPU time required by the standard DP algorithm.

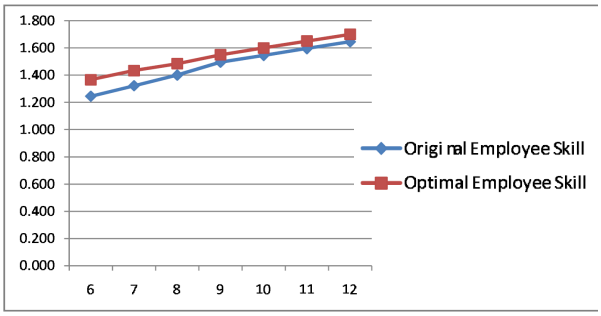
(a) Employee number



(b) Training spending



(c) Employee skill



(d) Service project capacity

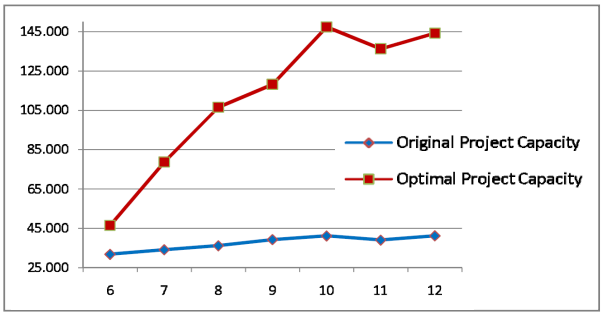
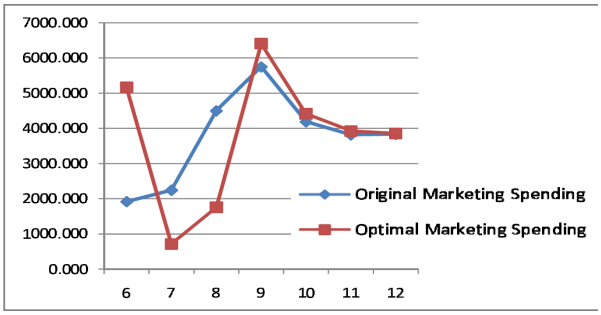
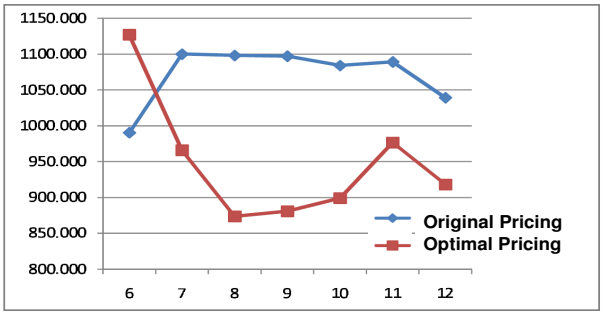


Fig. 3. Comparison of human resource operations.

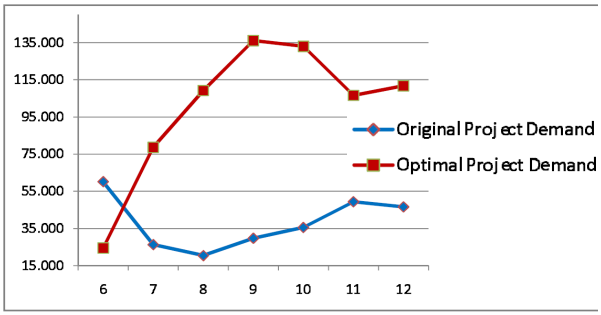
(a) Advertising expenses



(b) Project pricing



(c) Service project demand



(d) Service project projection

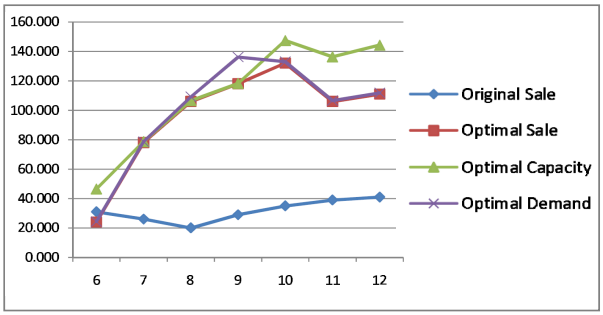


Fig. 4. Comparison of marketing operations.

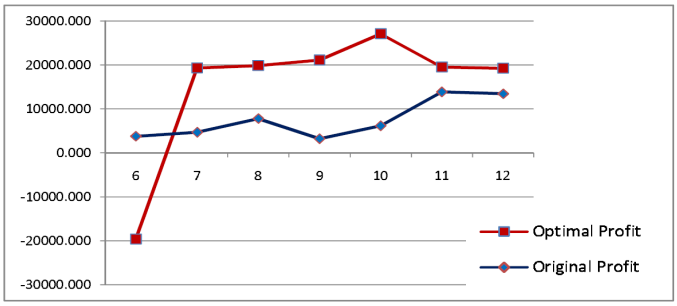


Fig. 5. Comparison of profit.

V. A Consulting Methodology Based on DMM4SE

In this section, we propose a consulting methodology based on our decision-making model for service enterprises (DMM4SE). Our methodology aims to coordinate and integrate different components of an service enterprise (e.g. manpower, asset, advertising, and pricing) so as to achieve centralized business goals. The proposed consulting methodology consists of the following steps (Fig. 6).

Step 1. Model training: In this step, historical operation data and related market information are collected. Given the past running information of an enterprise, each submodel in DMM4SE can be fitted as discussed in Section II. Sometimes service enterprises may not maintain a comprehensive historical data, and some enterprises may lack the evaluation system for abstract data such as advertising capital, employee skill-level, and asset value. Under these circumstances, the consulting team needs to mine the abstract data from other information sources. Furthermore, market information may not be accessible to public. To obtain pricing and advertising data of a specific market segment, the consulting team needs the support from third-party data service firms.

Step 2. Performance evaluation: Based on the fitted model, optimal decisions for the past periods are calculated by running the proposed ADP algorithm. A comparison between the actual and the optimal operations can provide evaluation of the past actions of the enterprise.

Step 3. Market analysis and forecast: Strategic planning of service enterprises relies on accurate forecasting of market demands. The service market can be highly dynamic. Different market sectors may behave differently, and even in the same market sector competitors may follow significantly different strategies. However, since we focus on strategic decision-making, market demands and competitor strategies can be aggregated and smoothed—this lowers the risk and uncertainty in forecasting.

Step 4. Goal definition: Competing enterprises may have different goals and consequently take different strategies. For example, three generic competitive strategies for service firms are proposed in [32]. They are *Overall Cost Leadership*, *Differentiation*, and *Focus*. A low-cost position relies on efficient-scale asset and skilled employees. Differentiation aims at customer loyalty and lies in creating novel services. The focus strategy rests on the premise that a firm can serve its narrow target market more effectively and efficiently.

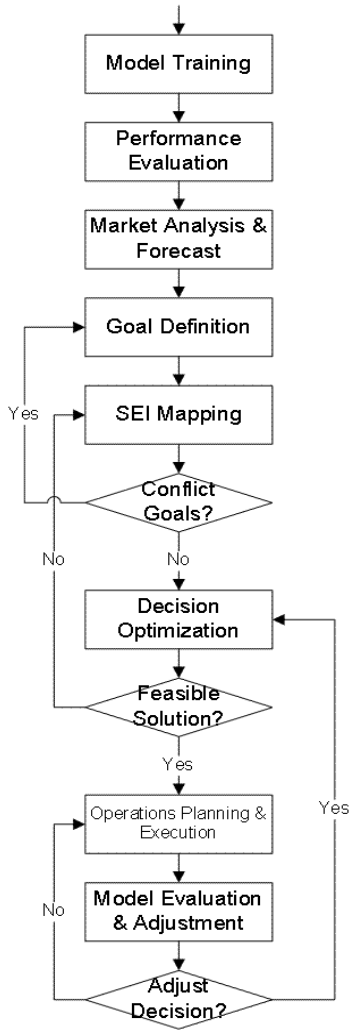


Fig. 6. DMM4SE-based consulting methodology.

Step 5. SEI mapping: By using the decision map (Fig. 7), each strategic goal can be mapped into one or more SEIs. Each SEI consists of four perspectives, i.e., finance, operations, marketing, and innovation, and each perspective is linked to the operation details. With this decision map, the target values of specific operation variables can be created. Sometimes, different goals are not compatible with each other. For example, to achieve a cost leadership, a service enterprise should reduce redundant resources such as idle employees and assets. However, to dominate a market segment and attract more customers, a service enterprise needs to respond to unexpected events such as peak demand. Therefore, a certain amount of redundant resources is necessary. Via SEI mapping, the consulting team identifies goal congruency, i.e., how well the goals match with each other. If some conflicts are not amendable, the consulting team has to go back to Step 4 and modify the goal definitions.

Step 6. Decision optimization: Based on the forecast of market demands and competitors' strategies, optimal decisions for the future periods are calculated using the proposed ADP algorithm. Since multiple constraints are added via SEI

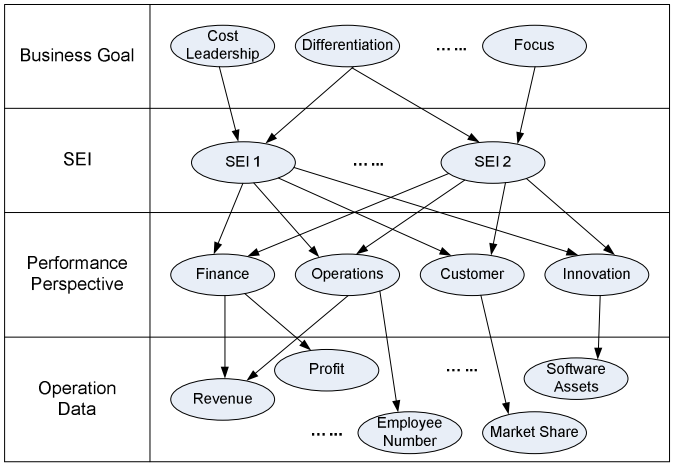


Fig. 7. Decision map of service business.

mapping, feasible solutions may not always exist. For those cases, the consulting team revisits Step 5 and adjusts the SEI mapping. If the SEI adjustment cannot reconcile the conflicts, the consulting team should go back to Step 4 and redesign the strategic goals. Generally, constraints can be categorized into different levels, from critical to optional. For instance, in a competitive market segment, demand is the most influential constraint. It is not easy to leverage the demand via pricing and advertising. Therefore, maintaining existing customers is the most essential task. However, in an emerging market segment, innovation is the differentiation power. Domain experts and superior assets are scarce resources in the industry. Therefore, the service capacity is the primary constraint. The priority of constraints can be incorporated into the decision model by designing corresponding penalty functions.

Step 7. Operations planning & execution: The DMM4SE focuses on decision-making in the strategic level. Each submodel in Section II is aggregated, summarized, and simplified to represent the major characteristics of the full model of the enterprise dynamics. Following the optimal decisions calculated for the full model, the enterprise operates its service business, subject to dynamic changes of the real business environment.

Step 8. Model evaluation and adjustment: At the end of each decision period, the service enterprise compares the realized results with the predicted results. If the realized results conform to the prediction, the model is validated and the calculated optimal decisions are reliable. If there is a significant difference between the realization and prediction, the consulting team needs to determine whether this is due to incorrect market forecasting or incorrect model. If the model is not accurate, it needs to be refined, and the consulting team should return to Step 3 and generate new optimal decisions. Besides the model evaluation, the service enterprise also needs to evaluate its business performance and identifies its position changes in the industry. If necessary, the strategy is adjusted to fit the dynamic service market.

VI. Discussion

A. Service Ecosystem

The proposed DMM4SE can be generalized to facilitate collaboration in a service ecosystem. In the service ecosystem, customers raise the requirement and pay for the services; service enterprises provide the services; and service-component providers focus on specific functionality that is reusable in a variety of industries and customers. Service business, unlike the traditional business, relies on the collaboration of all these "players" in the service ecosystem. Interaction between customers and enterprises determines the result of a service engagement. Different service enterprises may compete in some market segments but collaborate in other market segments to provide advanced solutions, which are beyond the resource or capability limits of individual enterprises. The service component providers also need to interact with clients so as to create service innovations. Therefore, how to manage partnership, outsourcing relationship, and parent-child relationship affects the decision-making of a service enterprise. Service enterprises have to consider their roles in the entire ecosystem in order to achieve optimal business performance. In our future work, we will incorporate in the DMM4SE the above mentioned collaboration among different components of the service ecosystem.

B. Prototype System

Using the newly released IBM Lotus Mashup Center [33], we can build a prototype system to deliver consulting services for service enterprises. Fig. 8 shows the architecture of the prototype system.

- Via BizRSS Feed [34], *Data Collector* retrieves operation data of a specific enterprise. By connecting to the public data portal, the data collector can obtain the finance and marketing information from the related industry. The attained data is saved in internal data warehouse for short-term and long-term analysis.
- *Learning Engine* analyzes the data and fits the parameters of the sub-models for manpower dynamics, asset capacity, advertising capital, and market demand. Then, the four sub-models are integrated into the complete decision-making model.
- Given the decision-making model, *Optimization Engine* runs the proposed ADP algorithm and output the optimal decisions.
- *Consulting Portal* provides for consultants a set of dashboards, which include SEI & Stock, human resource, market & customer, and project operations. These dashboards help consultants to prepare the business planning reports for service enterprises.

VII. Conclusion

The current service business lacks automated and quantitative tools guiding the operation and decision-making of service enterprises. To fill this gap, our study made the following contributions: First, we formulated the decision-making of a service enterprise as an optimal control problem. This allowed

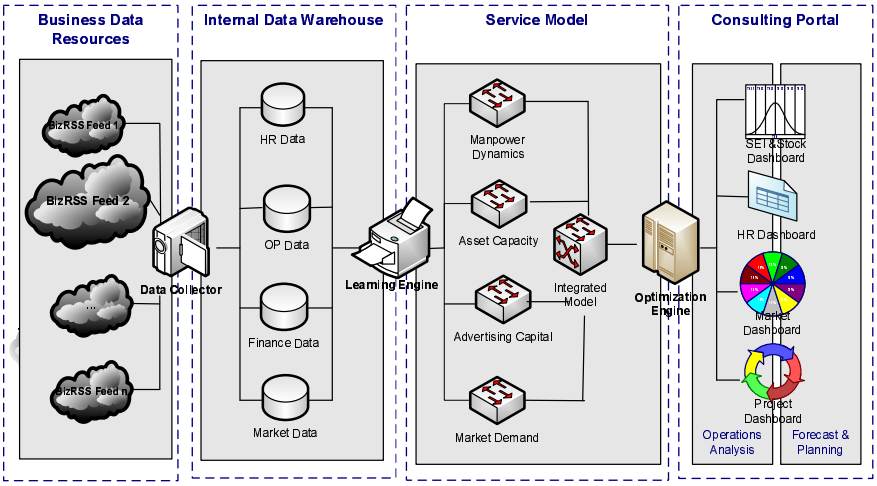


Fig. 8. Prototype system architecture.

mathematical description of service dynamics and facilitated quantitative analysis for the operation and strategic planning of service enterprises. Second, we proposed an approximate dynamic programming algorithm to solve the optimal control problem and successfully handled the model complexity by simplifying the computation of value functions using cubic spline interpolation. Third, we validated the optimal-control-based decision-making model in a simulation practice of the Beacon business. The results demonstrated the feasibility and effectiveness of the proposed model and solution approach. Finally, we introduced a DMM4SE-based consulting methodology and discussed its usages for service consulting practices.

However, the study presented in this paper has two major limitations: First, although the enterprise model developed here was a dynamic model (which characterized the dynamics of the enterprise over time), it was a time-invariant model—the model parameters were fixed and cannot be calibrated over time adaptively. Second, this study might have underestimated the competition from the other enterprises in the same market sector. The other enterprises may actively and intelligently adjust their strategies, so our "optimal" strategy developed based on prior observation of market behaviors may no longer be optimal in future decision-making.

To overcome the aforementioned limitations, we will introduce in future work a learning mechanism for model calibrating and tuning in order to capture the time-varying nature of the enterprise and the market. Moreover, in future work we will introduce game-theoretic components into our current framework of optimal control to address explicitly the competition and cooperation among the enterprises in the same market sector.

ACKNOWLEDGMENT

The authors would like to thank the editors and reviewers for their valuable suggestions and comments.

REFERENCES

[1] J. Spohrer, P. P. Maglio, J. Bailey, and D. Gruhl, "Steps toward a science of service systems," *IEEE Computer Magazine*, vol. 40, no. 1, pp. 71–77, Jan. 2007.

[2] Z. Kenessey, "The primary, secondary, tertiary and quaternary sectors of the economy," *Review of Income and Wealth*, vol. 33, no. 4, pp. 359–385, Dec. 1987.

[3] R. Qiu, Z. Fang, H. Shen, and M. Yu, "Editorial: towards service science, engineering and practice," *International Journal of Services Operations and Informatics*, vol. 2, no. 2, 2007.

[4] (2008) The Service Annual Survey, U.S. Census Bureau. [Online]. Available: http://www.census.gov/econ/www/ servmenu.html

[5] (2008) The 2008 World Factbook, U.S. Central Intelligence Agency. [Online]. Available: https://www.cia.gov/library/ publications/the-world-factbook/fields/2012.html

[6] B. Dietrich and T. Harrison, "Serving the services industry," *OR/MS Today*, vol. 33, no. 3, June 2006.

[7] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. New York & Beijing: Springer & Tsinghua University Press, 2007.

[8] J. Gharajedaghi, *Systems Thinking*, 2nd ed. New York: Butterworth-Heinemann, 2005.

[9] D. W. McDavid, "A standard for business architecture description," *IBM Systems Journal*, vol. 38, no. 1, pp. 12–31, 1999.

[10] M. R. Jaske, "Interfacing large interactive systems models and the user: the case of a beef cattle enterprise model for decision assistance," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, no. 12, pp. 848–858, Dec. 1977.

[11] J. K. Ostic and C. E. Cannon, "An introduction to enterprise modeling and simulation," Los Alamos National Laboratory, Los Alamos, New Mexico, Tech. Rep. LA-UR-96-3554, Sept. 1996.

[12] M. Harzallah, G. Berio, and F. Vernadat, "Analysis and modeling of individual competencies: toward better management of human resources," *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 36, no. 1, pp. 187–207, Jan. 2006.

[13] C. T. Leondes and R. H. F. Jackson, *Manufacturing and Automation Systems: Techniques and Technologies*. San Diego, CA: Academic Press, 1992.

[14] C. J. Petrie, "Introduction," in *Enterprise Integration Modeling: Proceedings of the First International Conference*, C. J. Petrie, Ed. Cambridge, MA: MIT Press, 1992, pp. 1–14.

[15] (2009) Function model. [Online]. Available: http://en.wikipedia.org/wiki/Functional_model

[16] J. L. Whitten, L. D. Bentley, and K. C. Dittman, *Systems Analysis and Design Methods*, 6th ed. McGraw-Hill Companies, Inc., 2004.

[17] R. Dorfman, "An economic interpretation of optimal control theory," *American Economic Review*, vol. 59, no. 5, pp. 817–831, Dec. 1969.

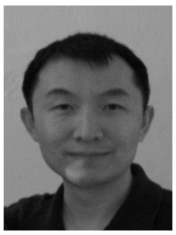
[18] L. Wu, R. D. Matta, and T. J. Lowe, "Updating a modular product: how to set time to market and component quality," *IEEE Transactions on Engineering Management*, vol. 56, no. 2, pp. 298–311, May 2009.

[19] D. Mendez and R. Narasimhan, "Examining market oriented aspects of cost of quality," *IEEE Transactions on Engineering Management*, vol. 49, no. 2, pp. 131–139, May 2002.
[20] Y. L. Tu, S. Q. Xie, and R. Y. K. Fung, "Product development cost estimation in mass customization," *IEEE Transactions on Engineering Management*, vol. 54, no. 1, pp. 29–40, Feb. 2007.
[21] C. Terwiesch and Y. Xu, "The copy-exactly ramp-up strategy: Trading-off learning with process change," *IEEE Transactions on Engineering Management*, vol. 54, no. 1, pp. 29–40, Feb. 2007.
[22] M. Nerlove and K. J. Arrow, "Optimal advertising policy under dynamic conditions," *Economica*, vol. 39, pp. 129–142, 1962.
[23] S. P. Sethi, "Dynamic optimal control models in advertising: a survey," *SIAM Review*, vol. 19, no. 4, pp. 685–725, Oct. 1977.
[24] O. Z. Aksin, "On valuing appreciating human assets in services," *Naval Research Logistics*, vol. 54, pp. 221–235, 2007.
[25] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, New Jersey: Wiley Interscience, 2007.
[26] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, New Jersey: Wiley Interscience, 2005.
[27] E. V. Shikin and A. I. Plis, *Handbook on Splines for the User*. New York: CRC Press, 1995.
[28] S. A. Johnson, C. A. Shoemaker, Y. Li, J. A. Tejada-Guibert, and J. R. Stedinger, "Numerical solution of continuous-state dynamic programs using linear and spline interpolation," *Operations Research*, vol. 41, no. 3, pp. 484–500, May-June 1993.
[29] M. A. Trick and S. E. Zin, "Spline approximations to value functions: a linear programming approach," *Macroeconomic Dynamics*, vol. 1, no. 1, pp. 255–277, Jan. 1997.
[30] (2005) Beacon Business Simulation, Nth Degree Systems Inc. [Online]. Available: http://www.nthdegreeonline.com/
[31] (2008) MATLAB, MathWorks. [Online]. Available: http://www.mathworks.com/products/matlab/
[32] M. E. Porter, *Competitive Strategy*. New York: Free Press, 1980, ch. 1, pp. 11–15.
[33] (2008) IBM Lotus Mashup Center. [Online]. Available: http://www-306.ibm.com/software/info/mashup-center/
[34] L.-J. Zhang, A. Allam, and C. A. Gonzales, "Service-oriented order-to-cash solution with business RSS information exchange framework," in *Proc. IEEE Internationl Conference on Web Services*, Chicago, IL, Sept. 2006, pp. 841–848.

Liang-Jie Zhang (M'99–SM'03) received the B.S. degree in electrical engineering from Xidian University, Xi'an, China, in 1990, the M.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, in 1992, and the Ph.D. degree in pattern recognition and intelligent control from Tsinghua University, Beijing, China, in 1996.

He is currently a research staff member and program manager of application architectures and realization at the IBM T. J. Watson Research Center, where he has made significant original contributions to services computing innovations and interactive media systems. He is the founding chair of IBM Research's Services Computing Professional Interest Community. He is the worldwide leader of IBM's SOMA Modeling Environment (SOMA-ME), which is the model-driven SOA (Service-Oriented Architecture) solution design platform for IBM. He is also the worldwide coleader of IBM's SOA Solution Stack (aka SOA Reference Architecture: Solution View) project. His new book, Services Computing, was published by Springer. He has received two IBM Outstanding Technical Achievement Awards, 10 IBM Plateau Invention Achievement Awards, an Outstanding Achievement Award from the World Academy of Sciences, and an Innovation Leadership Award from the Chinese Institute of Electronics. Dr. Zhang has been granted 36 patents and has 20 pending patent applications. As the lead inventor, he holds federated Web services discovery and dynamic services composition patents. He is the chair of the IEEE Computer Society Technical Committee on Services Computing. He is also the funding Editor-in-Chief of *IEEE Transactions on Services Computing*.

Zhe Shan (S'09) received the B.Sc. degree from Nanjing University, Jiangsu, China, in 2000 and the M.Phil. degree from City University of Hong Kong in 2003. He is currently a Ph.D. candidate in the Department of Supply Chain and Information Systems at the Pennsylvania State University. His major research interests include business process management, Web services, and service science. He is a student member of IEEE, AIS (Association for Information Systems), ACM (Association for Computing Machinery), and INFORMS (Institute for Operations Research and the Management Sciences).



Zhi-Hong Mao (S'96–M'01) received the dual B.S. degrees in automatic control and applied mathematics and the M.Eng. degree in intelligent control and pattern recognition from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, the S.M. degree in aeronautics and astronautics from Massachusetts Institute of Technology (MIT), Cambridge, in 2000, and the Ph.D. degree in electrical and medical engineering from the Harvard-MIT Division of Health Sciences and Technology, Cambridge, in 2005. He has been an Assistant Professor in the Department of Electrical and Computer Engineering and the Department of Bioengineering, University of Pittsburgh, Pittsburgh, PA, since 2005.

A Pattern Recognition Based Algorithm and Case Study for Clustering and Selecting Business Services

Liang-Jie Zhang, Shuxing Cheng, Carl K.Chang, and Qun Zhou

Abstract—Positioned as the backbone of service assets management console, a service registry has to enable real-time and offline service selection in an effective manner. This paper presents an analytic algorithm to guide the architectural design of services exploration in a services registry. Services assets are proposed to be framed into a well-established categorical structure based on pattern recognition algorithm. This design aims to provide systematic methodology and enablement architecture for analyzing, clustering and adapting heterogeneous services for dynamic application integration. The exploitation of pattern recognition algorithm maps a large amount of services into a manageable feature space, which consists of attributes that are related to static description and dynamic features, such as historical QoS and service-level agreement. The proposed architecture and associated services exploration methodology have been integrated into an industry strength SOA solution design platform. We also present a case study using the developed platform to illustrate the proposed algorithm for business services clustering and selection.

Index Terms— Pattern Recognition, Clustering, Service-Oriented-Architecture (SOA), QoS, Service Level Agreement (SLA), Probabilistic Estimation, Analytic Services Litmus Test (ASLT), SOA Solution Design.

1 Introduction

Service-oriented architecture (SOA) provides a flexible and extensible architecture to support dynamic business strategies and execution in the field of Services Computing [4][53]. SOA-based solution design approaches enable enterprises to deliver various types of services based on services assets that are either existing or newly created.

In order to be part of an SOA solution, a service provider needs to create its service interface and get them to be published to a service registry in the form of an electronic yellow page for service requesters to query and invoke. Service registries help to enable asset reuse in the context of SOA-based solution design [2][20][46]. In the near future, thousands of distinct service entities are expected to reside in various service registries. Moreover, a set of local service registries can be interconnected through federated services search engines [52]. With respect to a given service request, yielding a manageable search outcome from this large number of available service assets is a concern for practical SOA applications. In practice, there are various requirements from different lines of business in a client engagement setting. These requirements can differ in their respective priorities. Given a limited budget and delivery schedule, practitioners need a systematic and consistent way of prioritizing services based on various types of features including capability, availability, reusability, and cost. The current design of service registry in the market lacks a well-organized structure to help address feature-based service management and service-aware exploration. This is the major motivation underlying our proposal for a hierarchical service management framework to support efficient service exploration based on the requirement-oriented service feature set. Following our previous work [56], in this paper we detail a cascaded services exploration methodology in this paper, consisting of three steps: *services categorization*, *services clustering* and *services exposure*. This methodology aims to combine rigorous quantitative analysis and practical design expertise to support the hierarchical service asset management scheme for building an industry-strength end-to-end SOA solution design platform [1][59]. *Services categorization* introduces the SOA design efforts at the business level, which is typically conducted by business analysts in project teams. *Services clustering* group services into different sections based on a set of pre-defined features using pattern recognition algorithms. *Services exposure* refines the result of services clustering by employing the expert knowledge and user-specified constraints as input through a GUI based customization interface. At the end of the paper, a software prototype is presented to illustrate the proposed methodology.

The remainder of the paper is organized as follows. Section 2 gives a high level overview on multi-level services management schemes and reviews our previous work on this topic. Section 3 introduces the cascaded services exploration methodology and section 4 discusses

- *Liang-Jie Zhang is with IBM T.J. Watson Research Center. E-mail: zhanglj@us.ibm.com.*
- *Shuxing Cheng is with the Department of Computer Science, Iowa State University. E-mail: scheng@cs.iastate.edu.*
- *Carl K.Chang is with the Department of Computer Science, Iowa State University. E-mail:chang@cs.iastate.edu*
- *Qun Zhou is with IBM T.J. Watson Research Center. E-mail: qzhou@us.ibm.com*

the design details of pattern recognition based hierarchical services clustering algorithm. In section 5, we illustrate the proposed methodology for clustering and systematically selecting business services through a case study. We conclude our work with some research directions in section 6.

2 A Multi-Level Service Management Scheme

A generic business process is composed of several tasks to achieve a certain business objective. In service-oriented systems, each task is realized by a service provided by a service provider [31][53]. Fig.1 displays a service-oriented business process, which consists of six different service tasks. The branch "S3 (Marketing) → S5 (Sales)" constitutes the workflow of business management controlling the transactions of physical commodities represented by branch "S4 (Packaging) → S6 (Shipping)".

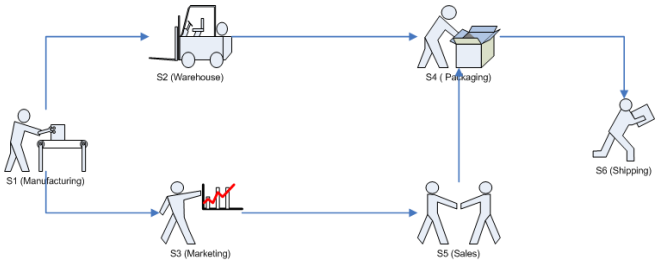


Fig.1. An example of service-oriented business process

2.1 The Relationship between Business Level Management and Service Level Management

Recent trends see more and more technology oriented enterprises employing well designed enterprise management systems to power their profit generation engines. This kind of enterprise management system works as localized management console at service level. Their deployments aim to provide better control over resources and operations within the realm of a single enterprise. For instance, data server management software concentrates on managing the performance metrics including response time and power usage level, ultimately reducing operation costs. On the other hand, the execution of a business process needs to cover the business interest spanning across different service providers. This is better understood from an instance like social welfare, which is a fundamental building block for economic theory [33]. Naturally, the business interest of a single service provider may not necessarily fit well to the social welfare represented by the goal of executing the associated business process. In a well behaved marketplace, we need to implement a centralized management station which aligns the social welfare at the business process level and the business interest at the service level, which is managed by localized service management console.

2.2 Centralized Business Explorer and Discovery Engine

In our previous work [51][56], we discussed the design of Business Explorer and Discovery Engine (hereafter denoted as BEDE), which performs as a centralized management console at business level. Its essential features include:

- Business-Service Alignment Adaptor: This module coordinates the business interests at different levels. There exists a bidirectional transformation between high-level business goals and lower-level service metrics of involved service providers. These transformations affect various operations throughout the business process execution lifecycle. For instance, decomposing the business goal to lower-level performance metrics helps determine which service provider to choose for a given service-oriented task. On the other hand, aggregating the lower-level performance metrics can help the system predict the final business performance.
- A Hierarchical Service Management Structure: This aims to manage a large amount of services that register in BEDE. In this paper, we are going to discuss a pattern recognition based algorithm to build this hierarchical structure.
- A Service Composition Engine: This engine is embedded with a set of optimization algorithms that is implemented based on the information provided by business-service alignment adaptors and the hierarchical service management structure to discover and aggregate several services into the targeted business process [55].

These components enable BEDE to coordinate the operations of distributed localized service management consoles, which also places BEDE as the interface between an outside service consumer and various autonomous service providers who do not directly interact with the consumer. Fig.2 illustrates the design of BEDE.

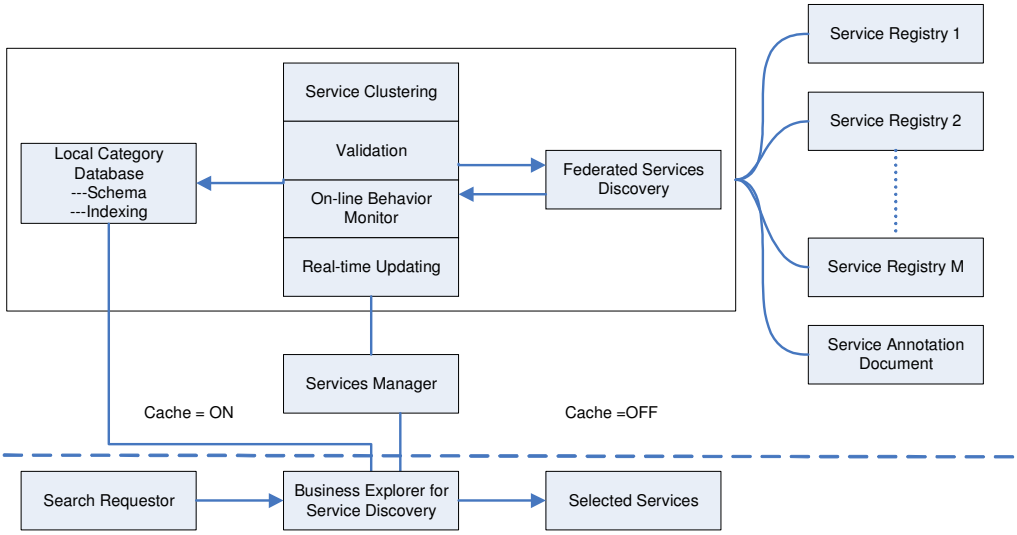


Fig.2. Architecturial Framework of BEDE

As shown in Fig.2, BEDE keeps track of various service providers through the service registry," which can be an instantiation of a UDDI standard or an enhanced version of UDDI, such as the WebSphere Service Registry and Repository [46]. Each registered service is identified with a service profile, treated as an integrated information model that encodes various aspects throughout the life-cycle of a service [2][36]. In reality, the service profile provides categorical information representing basic functionalities of the service. This type of categorical information constitutes the starting point for our research. The aggregation of these service registries can be queried through "Federated Services Discovery," which spans its search domain across a variety of service registries [52][57]. A service clustering module is the core of the proposed hierarchical service management structure. The clustering output guides the construction of a local category database, which transforms the randomly placed service registries into an organized structure. The service discovery process is performed on the Localized Category Database rather than on the unstructured service registries. This accelerates the querying process. Once the "Cache" is switched to "OFF," BEDE queries the candidate service directly from service registries through the Federated Services Discovery instead of the Local Category Database. The design of this switch will enable increased flexibility for the service asset manager.

2.3 Localized Service Management Console

Positioned as the decentralized counterpart to BEDE, the Localized Service Management Console (hereafter denoted as LSMC), deployed by each service provider, manages the local resources and collaborates with BEDE to manage the life cycle of a business process.

The business partnership between LSMC and BEDE is framed through the service level agreement (hereafter denoted as SLA) [19][29]. In our research, SLA is composed of a set of clauses agreed upon by both BEDE and the service provider. For instance, a typical SLA clause can specify that the shipping time should be within three days of receiving the request. A service level agreement configuration is represented as an (n+2)-tuple, which is

$$SLA = \{PM_1, \cdots, PM_i, \cdots, PM_n, SC, PE\} \quad (1)$$

In Eq.(1), PM_i represents the i-th stipulate clause, e.g., the quality value for a non-functional requirement asso-ciated with the delivered service. Service cost, identified as **"SC"**, is also a clause and represents the payment for consuming the service once the specified quality values are realized in the delivered service. The signed SLA represents a bidirectional obligation. The service consumer needs to pay the cost to the service provider, while the service provider holds the liability to service the consumer in compliance with the stipulated specifications. Failing to satisfy any specified non-functional requirement causes a penalty in terms of value of "**PE**" stipulated in SLA. Both BEDE and every connected service provider retain a copy of the signed SLA. Based on the parameters extracted from these stored SLAs, the service composition protocol launched by BEDE determines which service provider will be selected for a given service task. On the other hand, a given service provider needs to decide whether he can fulfill the service request allocated by BEDE in accordance with the specified quality values and its service capability, e.g., the estimated response time for a given task. This is one of the major responsibilities held by LSMC.

LSMC is composed of two collaborative functional modules: Decision Making Module and a Self-reconfiguration Module. The Decision Making Module (hereafter denoted as DMM) consists of four components: SLA Repository, Market Environment Monitor, Service Infrastructure Monitor and Service Access Controller. The Self-reconfiguration Module (hereafter denoted as SRM) is composed of two components: SLA Design Module and Service Infrastructure Maintenance Module. Their inter-

woven relationships are illustrated in Fig.3.

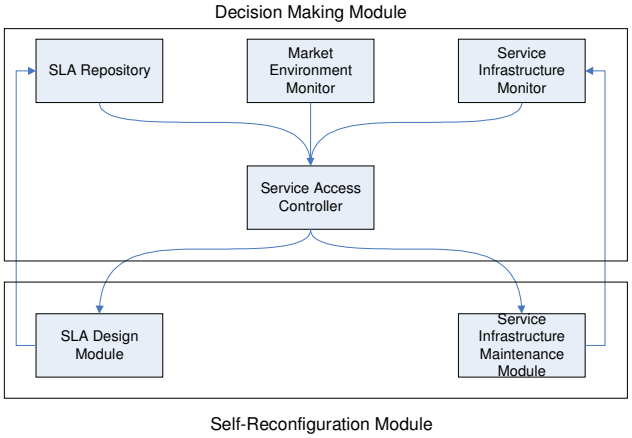


Figure.3. Architecture of DMM and SRM

Once the service provider receives the service request assigned by BEDE in accordance with the result of the service composition protocol, it will make decisions based on two factors: real-time system performance and market conditions. Real-time system performance is calculated based on the information collected by the Service Infrastructure Maintenance Module. If the system performance does not meet the enlisted SLA specifications, the Service Access Controller has to reject the request and pay the penalty stipulated in SLA. A quantitative based performance analysis suite is implemented on the Service Access Controller to compute a variety of performance metrics such as availability, throughput and response time. The core of this performance analysis suite is a queuing modeling tool [21]. In our previous work, we built queuing models to compute these metrics [10]. On the other hand, market conditions can affect providers' decisions as well. For instance, serving the request with high level quality can generate more revenue to the service provider. Therefore, a high demand on high quality services will direct the service provider towards shifting his resources to the most profitable requests. In the design of DMM, the Market Environment Monitor keeps collecting various types of market information that a service provider can use to increase profitability.

3 Cascaded Service Exploration

From the role playing perspective, BEDE collects, manages, composes and finally delivers the required business process to the customer. These functionalities of BEDE are realized through Cascaded Service Exploration (hereafter denoted as CSE), which represents a systematic methodology for organizing the large number of services registering with BEDE. CSE was first proposed in [56], which is composed of three critical steps:

- services categorization
- feature-based multi-level service hierarchy construction and management
- hierarchy-based service discovery

In a service registry, every published service is annotated with categorical information stipulating the basic functionality that this service delivers. For instance, service A is categorized as a shipping service. Based on this type of categorical information, the registered service items can be organized into different categories, which provide a preliminary grouping structure to refine the original unstructured service registry. Nevertheless, each category can still contain too many service items, and thus the category-based service registry is not a manageable solution in terms of the size of a single category.

In practice, two services coming from the same category can still differ a lot in accordance with the collection of features used to characterize them. Consider three registered services, A, B and C, which all belong to the same category. Service A is more similar to service B than C in terms of reliability; however, buying service B costs the same as buying service C, while it is much cheaper to buy service A. This example shows that a set of services can illustrate different grouping structures depending on different evaluation criteria.

The second step of CSE, feature-based multi-level service hierarchy construction and management, aims to provide a fine-grained and scalable solution for the further refinement of the service category. In the next section, we discuss the feature-based service quantification. Feature-based service quantification enables us to apply pattern recognition techniques and cluster services in accordance with various types of service attributes. Furthermore, we extend the single clustering operation into a hierarchical services clustering operation, which results in a multi-level service hierarchy. In the context of CSE, the service discovery process is performed against the constructed hierarchy rather than over the whole service category. We discuss this in detail in section 4.6.

4 Business Driven Service Clustering Based on Pattern Recognition

Clustering algorithms have been widely used in a variety of application domains. They are used to manage a large amount of data and discover hidden relationships by grouping a collection of objects into several clusters based on a set of given features. A generic clustering analysis is conducted through the following steps:

1. **Feature selection**: In this step, a set of features is determined to characterize the studied objects. The focus of feature selection is to build a manageable feature space that can be used to separate every studied object as well as to maintain a satisfactory level of computation cost [18] [39].
2. **Distance measure selection**: This step is to decide a quantitative measurement that counts the proximity and dissimilarity among different points in the constructed feature space. Potential candidates for the distance measure include Euclidean distance, correlations and mutual information [18] [23].
3. **Clustering procedure**: A specific clustering algorithm is chosen to build clusters in the related fea-

ture space based on the distance measure chosen in step 2 [18] [48].

Given the same set of objects, the final clustering result differs in accordance with the combination of features, distance measure, and the selected clustering algorithm.

4.1 Building Feature Space for Service Systems

In the SOA solution design, each service profile contains a collection of features spanning the whole spectrum of service attributes, ranging from business oriented attributes to system centric ones. They can be basic functionality descriptions, such as "shipping service," or non-functional characteristics, such as "service availability." In the SOA solution design, we tend to take into account the following service features, which usually play critical roles in characterizing services based on practical SOA design experiences [4][20][42][43].

- **Service Accessibility**: quantifies the successful invocation rate of a service instantiation during a given time interval. Services accessibility can vary with the volume of services requests, making it an applicable metric for evaluating system scalability.
- **Service Cost**: the price tag that is charged by the service provider, which is the sum of the actual cost of rendering the service and the net profit stipulated by service provider.
- **Service Interface**: defines the gateway for communicating with other services, which includes services data exchange format and supporting communication protocols, etc. This information is essential for the service composition process.
- **Service Reliability**: measures the ability of a service provider to satisfactorily deliver service in compliance with the enlisted functional and non-functional requirements in a given time period.
- **Service Response Time**: equals the elapsed time between the time point of receiving a service request and the time point of finishing service delivery.
- **Service State**: used to capture the service dynamics over its lifecycle. For instance, the amount of usage for a stock quote service is represented as a state. Its inherent randomness can be modeled by either a discrete or a continuous stochastic process.

Every selected feature is assigned with a numerical range over which we can score a particular service. The aggregation of these numerical ranges constitutes the feature space where each service is represented as a point from the geometric point of view. To illustrate an operational example for quantifying service reliability, we count the number of total requests in a given period, denoted as R_t. The number of failed response in the same period is referred to as R_f. Naturally, $R_f \leq R_t$. The service reliability (hereafter denoted as SR) can be computed as $SR=(R_t - R_f)/R_t$. We quantify service reliability in a "1-5" scale. A higher SR score stands for a more reliable system.

TABLE 1
QUANTIFICATION PROCEDURE FOR SERVICE RELIABILITY

Range of Value of SR	SR score
$SR \geq 0.9$	5
$0.8 \leq SR < 0.9$	4
$0.7 \leq SR < 0.8$	3
$0.5 \leq SR < 0.7$	2
$SR < 0.5$	1

This kind of quantifying procedure differs in accordance with the particular feature being studied [6] [9]. For instance, with respect to service response time, we can compare the advertised response time with a benchmark response time extracted from a stored service history record or through third party auditors. This comparison result can be used to evaluate the performance level of a service provider, again scored on a numerical scale of "1-5." Subjective factors, which come from the SOA solution designer or other users, can also affect the quantifying procedure. In the above example of quantifying service reliability, the mapping between a numerical range and the associated score is decided based on either experience or system requirements.

4.2 Quantifying the Differences between Services

From the topological point of view, the feature set construction and the subsequent quantifying procedures serve to map the combined effect of quality aspects of each service onto a point in a high-dimensional feature space. Hereafter, we refer to this point as the **service point.** The relationships between different services are thus incarnated through the geometric structure composed by those service points. Given two services and their respective features, differences in terms of the combinations of these features are quantified by the distance between their associated service points. A typical distance measure in the geometric space is Euclidean distance, which is one of the most widely used distance measures in multivariate statistical analysis.

Selecting the distance measure is the first step for a clustering algorithm. Considerable research has been conducted to investigate the relevant metric for quantifying the similarities or dissimilarities. Different distance measures vary in their respective capabilities of capturing the embedded structure of the studied data [7][18][40]. Besides the widely used Euclidean distances, other distance measures, such as Perason correlation coefficient and mutual information, are also used [23][26]. For instance, mutual information-based distance measure is robust to missing data values, which is an expected feature for biomedical applications [16]. Beyond these popular distance measures, some researchers also report the specifically designed distance functions by taking into account the conceptual domain knowledge and underlying probabilistic models [12].

In this research, we use the Euclidean distance due to its popularity in the clustering-related applications.

$$D(S_i, S_j) = \sqrt{\sum_{l=1}^{n} (i[l] - j[l])^2} \qquad (2)$$

In Eq.2, S_i and S_j represent two service points respectively; $i[l]$ represents the numerical value for the l-th feature of point S_i; n represents the number of features used for characterizing a service. A service cluster is composed of the service points that have smaller distances from each other while they display larger distances with other service points in the given feature space. Each cluster C_a is represented by a functional centroid denoted as U_a, which is represented by a set of computed feature values, i.e., $U_a = \{U_{a,1}, \cdots, U_{a,t}, \cdots, U_{a,n}\}$. The definition of $U_{a,t}$ is given in Eq.(3).

$$U_{a,t} = f(S_1[t], \cdots, S_i[t], \cdots, S_n[t]) \qquad (3)$$

In Eq.(3), $U_{a,t}$ represents the numerical value for the t-th feature of U_a and is determined by the t-th feature values of all the service points belonging to this cluster. $S_i[t]$ represents the t-th feature value for the i-th service points belonging to C_a. $f(\cdot)$ stands for a function that computes the centroid, whose exact formula is task-dependent and is decided by the SOA solution designers. Eq.(4) illustrates an average-based functional centroid.

$$U_{a,t} = \frac{\sum_{L=1}^{M} S_i[t]}{M} \qquad (4)$$

In Eq.(4), $M = \|C_a\|$ represents the number of service points included in cluster C_a.

4.3 Services Clustering Algorithm

We use K-means clustering algorithm to conduct the clustering operation, which is listed in Table 2 [7][18]. From the viewpoint of practical applications, K-means algorithm achieves a reasonable balance between implementation simplicity and the clustering performance.

TABLE 2
K-MEANS ALGORITHM FOR SERVICES CLUSTERING

K-Means (M, K)
- M is the number of service points to be clustered
- K is the number of clusters

```
1:  Randomly select K service points to represent the
    centroids for the K clusters denoted as C_i: i=1,…,K;
2:  PER = 0; Iter = 1; H=0;
3:  WHILE ( H > • ) or (Iter < N )
4:      FOR j = 1 To M
5:          Assign S_j to the C_i whose functional centro-
            -id has the smallest distance with S_j;
6:      ENDFOR
7:      FOR i= 1 to K
8:          The functional centroid of C_i is
            computed based on the service points
            assigned to C_i;
9:      ENDFOR
10:         PER_1 = Σ_{i=1}^{K} Σ_{j=1}^{M} ||S_j − C_i||
11:     H = abs( PER_1 - PER)
12:     PER = PER_1 ;
13:     Iter = Iter +1 ;
14: ENDWHILE
```

In the K-means clustering algorithm, K is a predetermined parameter, which stipulates the number of clusters to be constructed. The aggregated distance between every service point and its belonged cluster, computed in Line 10, is used as an optimization objective. The optimization process will not stop until either the objective difference between two sequential iterations is smaller than a given threshold or the number of iterations surpasses the given value. This condition is reflected in Line 3. For each service point, the output of K-Means algorithm will stipulate the specific cluster that this service point belongs to.

4.4 Feature-based Multi-level Service Hierarchy

With respect to a given service category, we can construct a set of feature selections, denoted as $F = \{F_1, \ldots, F_i, \ldots F_n\}$, where F_i represents a feature selection and is used to constitute a feature space against which a clustering operation can be applied. Note that F_i can contain multiple features. A series of hierarchical clustering operations can be conducted based on F and constitutes a multi-level service hierarchy. Fig.4 illustrates a three level service hierarchy built upon a set of given feature selections.

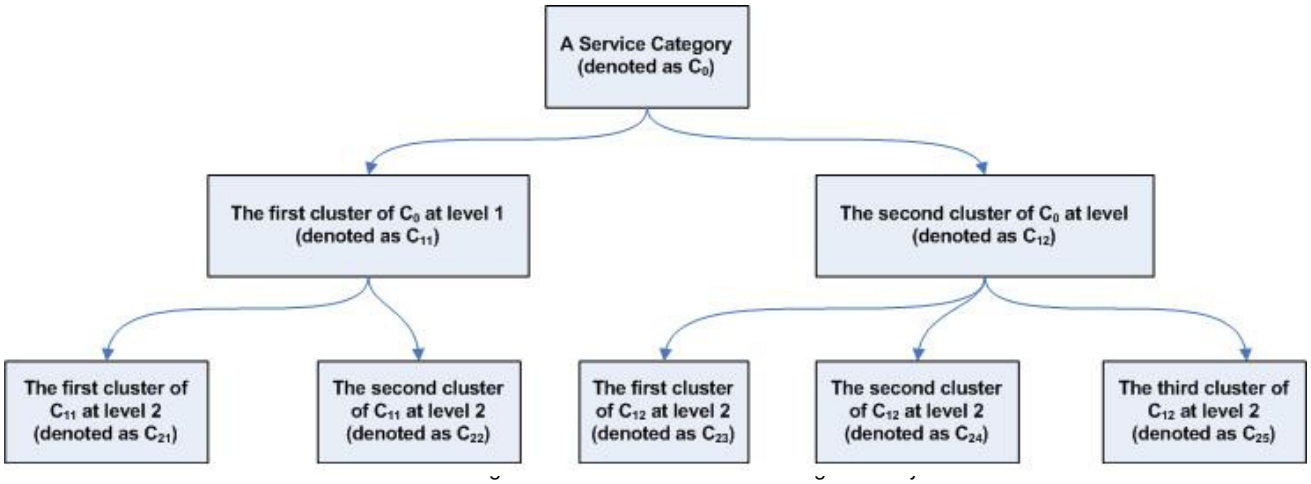


Fig. 4. Multi-level service hierarchy

The service hierarchy shown in Fig.4 is constructed for service category C_0, which is setup as the default root node. For notational purposes, we denote the level of C_0 as level 0. Each node represents a service cluster, and a level-i cluster represents the result of applying clustering operation against its parent cluster at level i-1 using feature selection F_i. Hence, a level-i cluster is also referred to as F_i-based cluster. Similarly, F_{i+1} is to support the clustering operation performed against the the F_i-based cluster and creates the F_{i+1}-based clusters. For instance, C_{11} and C_{12} share the same parent cluster, C_0. They are created by a clustering operation applied against C_0 using feature selection F_1. The subscript x of cluster C_x represents the global identification for this cluster.

For illustration purposes, we relate the multi-level service hierarchy, shown in Fig.3, with the hierarchical services clustering operation against a shipping service category. The root node is a category containing all of the registered shipping services. We organize this category into a more refined structure based on a set of selected features. The first feature selection, F_1, corresponds to the *transportation method*. This clustering operation results in two level one clusters: the airline-based shipping services are put into cluster C_{11}, while the ground shipping services are put into C_{12}. F_2, a collection of performance related features, such as shipping time and shipping safety, can be used to further refine the level 1 cluster if its size is still too big for a manageable discovery process. This is reflected as five level 2 clusters shown in Fig.4.

4.5 Constructing and Managing the Multi-level Service Hierarchy

Multi-level service hierarchy is constructed by performing a set of hierarchical clustering operations over a collection of feature selections. The procedure for the hierarchy construction is listed in Table 3. By calling **HierarchyConstruction**(C_0, F, n), we can construct multi-level service hierarchy for the service category C_0 based on a set of feature selections, F= {F_1,…,F_i,…F_n}.

TABLE 3
CONSTRUCT HIERARCH FOR SERVICE CATEGORY C_0

HierarchyConstruction(C_0, F, n)

```
1:  FOR i = 1 TO n
2:     Initialize CL_i as an empty linked list
3:     cPtr = CL_{i-1}.head
4:     WHILE cPtr->next is not NULL
5:        CI← Cluster(cPtr, F_i)
6:        cPtr->child = CI
7:        Insert CI to CL_i
8:        cPtr = cPtr->next
9:     END WHILE
10: END FOR
```

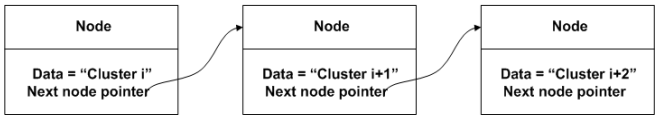


Fig. 5. Linked list for storing cluster pointers

In our implementation, we use linked list to storing the pointers that point to the cluster. The set of clusters locating in the same level is stored and managed by a single linked list. In Table 3, CL_i is a linked list, where each node stores the pointer pointing to a level-i cluster. "CL_i.head" stands for the header pointer of CL_i and cPtr stands for a node while traversing the linked list. Basically, the **HierarchyConstruction** iteratively goes through the feature selection vector F. The feature selection F_i is used to cluster every level-(i-1) cluster, represented by cPtr during traversing the linked list CL_{i-1}. In line 6, the generated

child clusters, CI, are connected to the parent node, cPtr; in line 7, all of the newly generated level-i clusters are connected via standard linked list attaching operations. The computation cost of **HierarchyConstruction** is measured in terms of two factors: (1) the size of F, which is n; (2) at each iteration, the algorithm will invoke the clustering operation, which is denoted as **Cluster** in line 5; the number of invocations depends on the size of linked list CL_{i-1}, which equals to the number of level-(i-1) clusters. Generally, the exact number of clusters obtained after performing a clustering operation depends on the clustering algorithm being used and the size of input set. The K-means clustering algorithm is unique since the number of generated clusters is pre-determined. However, this property is not shared by most of the clustering algorithms.

The constructed service hierarchy also needs to support the update operation. For instance, a newly-registered service S is required to be inserted into the service hierarchy. Table 4 lists a recursive-based update operation. The major task of hierarchy update is to traverse through the hierarchy and find the cluster at each level which shares the most similarity with S in accordance with the feature selection corresponding to that specific level. Here, S represents the service either to be inserted or to be deleted. The update operation is conducted by calling **HierarchyUpdate**(C_0, S, 0).

TABLE 4
UPDATE THE SERVICE HIERARCHY

HierarchyUpdate(C_x, S, i)
- C_x stands for the cluster that is being examined in the current recursive call
- S is the service that invokes the update
- i denotes the level where C_x locates

1: C_v←**SearchCluster**(C_x->child, S, F_{i+1})
2: **UpdateCluster**(C_y, S)
3: **IF** C_v->child = Null **Then**
4: Return
5**:** **ELSE**
6: Call **HierarchyUpdate**(C_v->child, S, i+1)
7: **END IF**

In the line 1 of Table.5, a level-(i+1) cluster sharing the highest similarity with service S in accordance with F_{i+1}, the feature selections used to create level-(i+1) clusters, is discovered in procedure **SearchCluster**. Line 2 is to update the discovered cluster through either inserting S or deleting S. The procedure of **UpdateCluster** is listed in Table 6. Line 6 is to traverse to the next level once the determined cluster, C_y, has a child, which is indicated by the condition of $i \neq n$. The complexity of procedure **HierarchyUpdate** depends on three factors: (1) the height of hierarchy, denoted as N_h; (2) the procedure **SearchCluster**, which will be invoked at each level and the realized complexity depends on the number of child clusters of C_x, and we can denote its upper bound as M; (3) the procedure **UpdateCluster,** which will be invoked at each level and the realized complexity depends on the size of selected cluster and we use Q to stand for the related upper bound. Hence, the total complexity of HierachyInsertion is $O(N_h \times (M+Q))$.

TABLE 5
THE PROCEDURE FOR FINDING THE CLUSTER THAT MATCHES SERVICE S BASED ON A GIVEN FEATURE SELECTION

SearchCluster(CQ_i, S, F_i)
- CQ_i is a pointer array where each entry points to a cluster
- S is the service to be searched
- F_i represents the feature selection to be used for comparing the similarity

1: a = **argmin** (**Dist** (CQ_i[b], S | F_i)) $\forall$ b =1,…,n
2: **Return** CQ_i[a]

In the service hierarchy, a parent cluster can have multiple child clusters, which constitute a set of sibling clusters. In Fig.4, C_{21} and C_{22} are two sibling clusters. The procedure **SearchCluster** is to search among a set of sibling clusters and return a cluster sharing the most similarity with S. The pointer array CQ_i points to a set of sibling clusters. **Dist** (CQ_i[b], S | F_i) is to compute the distance between service S and cluster CQ_i[b] in terms of the feature selection F_i.

The procedure **UpdateCluster** constitutes another essential operation for **HierarchyUpdate,** and is listed in Table 6. Updating a give cluster includes two tasks: one is to update the centroid of the cluster; the other is to update the service list being assigned to this cluster. In the implementation, a specific data structure, such as linked list, can be used to store the services for a given cluster.

TABLE 6
UPDATE A CLUSTER BY INSERTING/DELETING SERVICE S

UpdateCluster (C_x, S)
- C_x stands for the cluster to be updated by S

1: Update the centroid of C_x based on whether inserting S or deleting S
2: **If** inserting S **Then**
3: Attach S to C_x
4: **ELSE If** Deleting S **Then**
5**:** Remove S from C_x
6: **END IF**

4.6 Service Hierarchy-based Service Exploration

The third step of CSE is service hierarchy-based service exploration. In a service category lacking an organized services enlisting structure, a global discovery process has to be launched among all the candidate services for a given service request. With the help of introducing the multi-level service hierarchy to manage the service registry, we can traverse down the service hierarchy to find the cluster which best fits the request rather than directly searching for the best fit from all of the candidate services. After discovering the cluster which best fits to the request, we

can limit the searching domain to this cluster rather than the whole category for the final exploration. Table 7 lists the algorithm for discovering the cluster that best matches the service request.

TABLE 7
SERVICE HIERARCHY-BASED SERVICE EXPLORATION

HierarchyExploration(C_x, S, i)
- C_x stands for the cluster that is being examined in the current recursive call
- S is the service request
- i denotes the level where C_x locates

```
1: C_v ← SearchCluster(C_x->child, S, F_{i+1})
2: IF C_v->child = Null Then
3:     Return Cluster C_v
4: ELSE
5:     Call HierarchyExploration(C_v->child,S, i+1)
6: ENDIF
```

By calling **HierarchyExploration**(C_0,S,0), this recursive-based procedure outputs a cluster C_y whose functional centroid shares the most similarity with the service request. Hence, we narrow the search domain from a category to a cluster. Note that the traversing process of **HierarchyExploration** is similar to the one of **HierarchyUpdate** listed in Table.4. However, the operation of **UpdateCluster** invoked at each level is not needed in **HiearchyExploration.** Hence**, HierarchyExploration** requires less computation cost**.** In the end, the searching operation is performed against the cluster returned by **HierarchyExploration** to decide the service needed to be delivered for serving the request. The associated computation cost is determined by the size of the cluster.

5 CASE STUDY

In this section, we discuss a software prototype that employs the CSE methodology for service discovery. The developed prototype, Analytic Services Litmus Test (ASLT) toolkit, has been realized in an end-to-end SOA solution design platform, SOMA-ME, which supports SOMA, a methodology developed by IBM aiming to deliver industry strength SOA design solutions [1][58]. To help the SOA solution designer, ASLT toolkit has several design features:

- GUI-based application that allows the SOA solution engineer to launch the underlying services clustering engine;
- GUI-based application that facilitates the input of expert knowledge;
- Context-aware menu that eliminates ambiguities in the SOA design.

The following case study illustrates how the CSE guided service discovery lifecycle is instantiated through the ASLT toolkit.

5.1 Service Categorization in ASLT Toolkit

For illustration purposes, we are considering a market composed of various types of service providers, including:

- Financial institutions that provide payment management services, such as traditional banks or online payment services;
- Customer management units, which provide various types of customer support services like opening a user account on the Web;
- Shipping service carriers, represented by specific service brands.

These different service providers can belong to the same enterprise or can be outsourced to a third party provider. After collecting all the registered services through the module of Federated Services Discovery, ASLT organizes these services providers into different categories based on their basic functionalities.

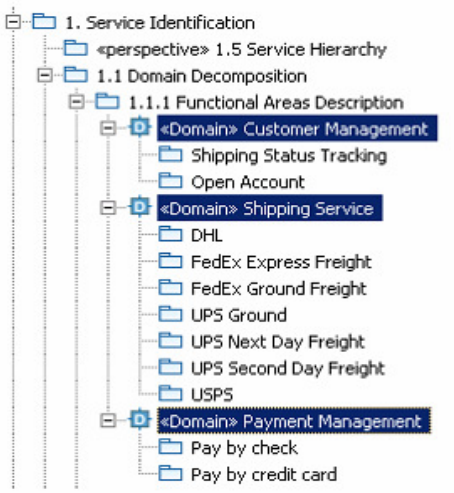


Fig. 6. Service categorization output of SLT

As shown in Fig.6, the studied example has three categories. The context notation of "Functional Areas Description" indicates that every category differs from others in terms of its basic functionality. Here, we have three types of functionalities: "Customer Management," "Shipping Service" and "Payment Management." In alignment with the notation used in SOMA-ME, we use "Domain" in our tool, which has the same physical meaning as "Category." Referring to the multi-level service hierarchy shown in Fig.4, three level-0 root nodes represent three different categories respectively.

5.2 Service Features Building Process in ASLT Toolkit

In the case study, we focus on the shipping service category to illustrate the hierarchical services clustering. Within the context of this study, three essential features—shipping cost, shipping time and shipping safety—are used to characterize a given shipping service. In the developed toolkit, SOA solution engineers launch the feature selection step using a context aware menu shown in Fig.7.

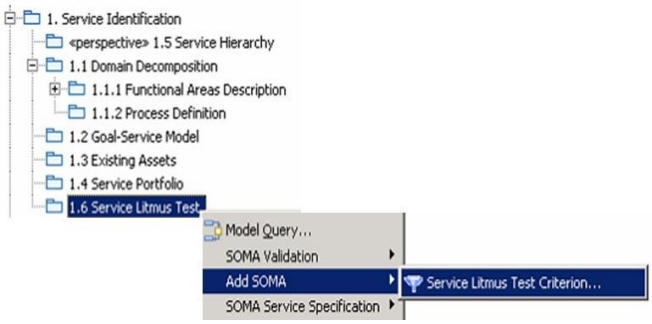


Fig.7. Launch the SLT operation through context guided menu

The ASLT tool provides a GUI-based information collection portal, shown in Fig.8, to facilitate the SOA solution engineers to input four types of information affiliated with each feature. The context information includes name of feature and necessary descriptions. This information is used for clarification and documentation purposes. Note that the feature name is labeled as "Name of Criterion" in the ASLT toolkit, which fits the technical term standard executed by internal development efforts. It is interchangeable with "Name of Feature". The items of "Minimum Value" and "Maximum Value" determine the numeric scale when we quantify this feature. In table 1, we discussed the quantification procedure for service reliability, where the maximum value is 5 and the minimum value is 1.

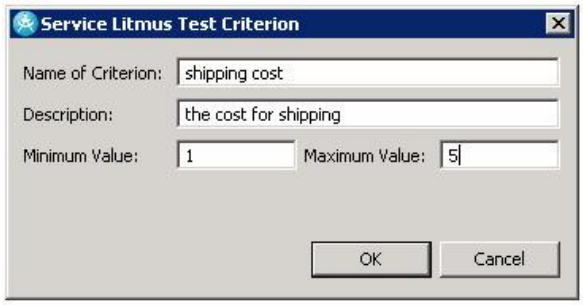


Fig. 8. Specify the feature of "shipping cost"

In this study, besides the feature of shipping cost, the shipping time and shipping safety are also selected as features for characterizing the studied services. The similar scoring processes are launched to specify the numerical ranges for these two features. The aggregation of the numerical ranges for all of these features determines the numeric sphere of feature space to be built. After the specification process, ASLT toolkit automatically puts these specified features under the subdirectory of "Service Litmus Test for clarification," which is illustrated in Fig.9.

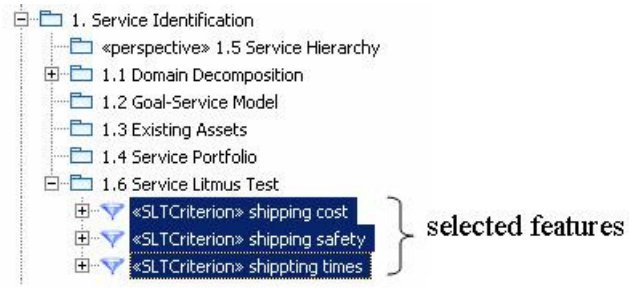


Fig. 9. Organized result for selected features

5.3 Quantifying Services and Clustering Procedure in ASLT

As shown in Table.1, with respect to given feature, each service is scored based on a predetermined scale. As a result, each candidate service is mapped to a service point in the built feature space. In the ASLT toolkit, SOA solution engineers use the GUI-based quantification tool to graphically score each candidate service. For instance, with respect to "UPS Next Day Freight," its shipping cost is below average on the scale of 1-5; its fast delivery schedule gives it a high score in terms of shipping time; its shipping safety, while not the best, is still above the average. The quantification procedure for this service is shown in Fig.10.

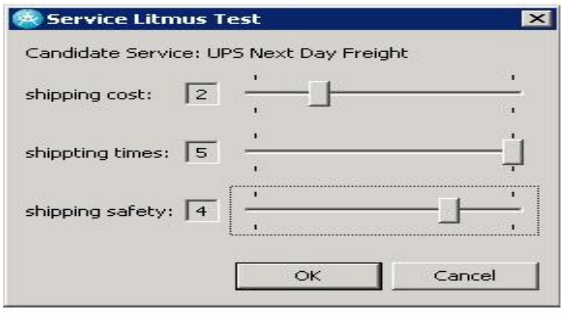


Fig.10. Quantify the service "UPS Next Day Freight"

This quantification procedure transforms the service "UPS Next Day Freight" into a service point in the feature space with coordinate (2, 5, 4). Similarly, we can conduct the quantification operation on all other five services belonging to the category of "Shipping Service." These transformed service points are shown in Fig.11, which displays a clear topological pattern among these six services. Four of them are near with each other while the other two can be clustered into a separate group.

Fig.1

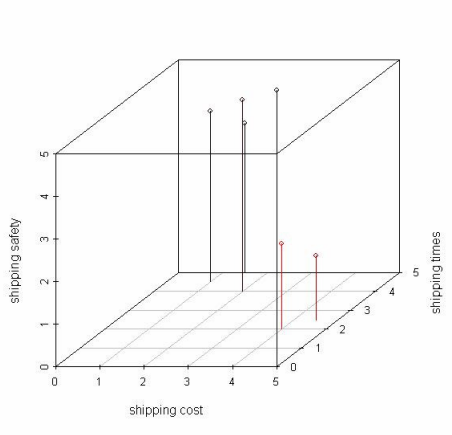


Fig.11. 3-D plot for the service points in the built feature space

In the current version of the ASLT toolkit, we implement K-means algorithm to perform the clustering operation. In the next phase of the development, a collection of clustering algorithms will be built to allow the user to select the one that best fits the problem at hand. Fig .12 illustrates the clustering result after running the clustering module.

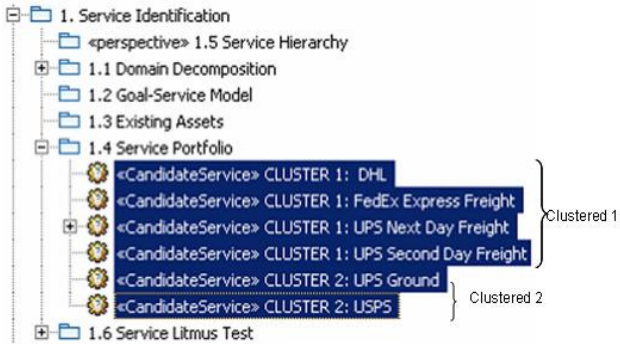


Fig.12. The output of clustering result

In the ASLT, the clustering result is placed under the directory of "Service Portfolio," which specifies a service cluster for each listing. In this example, the clustering operation builds a two-level services cluster hierarchy. Cluster 1 tends to include service providers which offer fast delivery services that charge more, while the cluster 2 provides budget constrained shipping services.

5.4 Service Discovery and Exposure in ASLT

In the above, cluster 1 and cluster 2 are generated based on the feature set of "shipping cost", "shipping time" and "shipping safety". Assume there has a request quantified as {"shipping cost=3", "shipping times= 4.5", "shipping safety=4.5"} in terms of the scale and feature set used to quantify the candidate services. The high score of both "shipping time" and "shipping safety" reflect that this customer demands a shipping service of high quality. The shipping cost of 3 implies that this customer tends to be neutral from the budget point of view. With respect to the given request, ASLT invokes the operation of **HierarchyExploration** to find the cluster which matches the request the best. Here, cluster 1 is the best fit.

Although the clustering operation can alleviate the computation effort of services discovery by narrowing down the search domain to a specific cluster, the existence of multiple candidate services belonging to the selected cluster requires additional screening processes.

With respect to this illustrative example, there are only four candidate services in cluster 1. We assume the target customer is a small business owner who intends to reduce the operational cost. Therefore, an exposure criterion is the amount of the discount offered to the small business owners. Note that the criterion itself is also a feature used to characterize a service. Fig.13 illustrates how the solution engineer sets up the exposure criterion in ASLT toolkit, which is a similar process of specifying service features shown in figures 8-a, 8-b and 8-c.

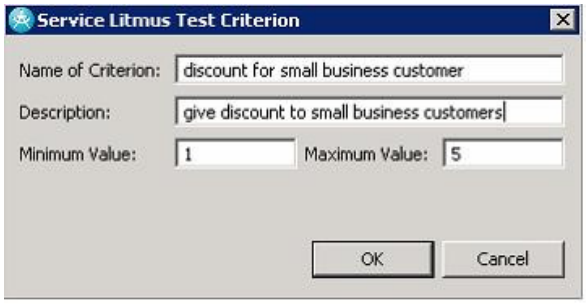


Fig.13. Specifies the feature to refine the cluster

In ASLT toolkit, the ideal service to be exposed is the one whose score passes the so-called "exposure threshold," which is set up through a GUI interface, shown in Fig.14.

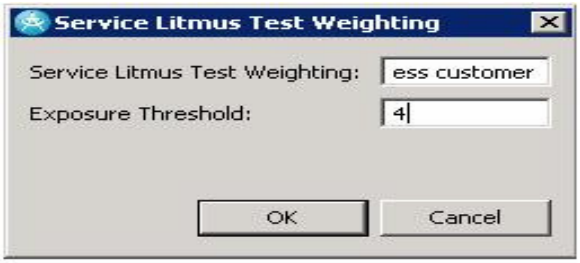


Fig.14. Setup the exposure criterion

In Fig.14, the term of "Service Litmus Test Weighting" is used to describe the exposure criterion, and the exposure threshold of 4 indicates that service being scored higher than 4 will get exposed. Figure 15-a, 15-b, 15-c and 15-d illustrate the scoring process for the candidate services.

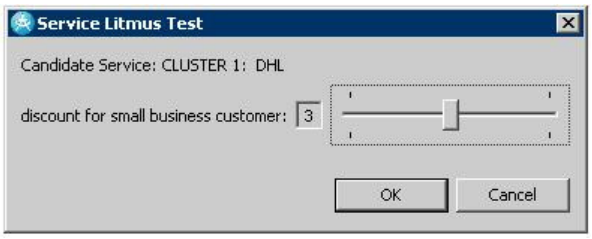


Fig.15-a. Scoring the service of "DHL"

Fig.15-b. Scoring the service of "FedEx Express Freight"

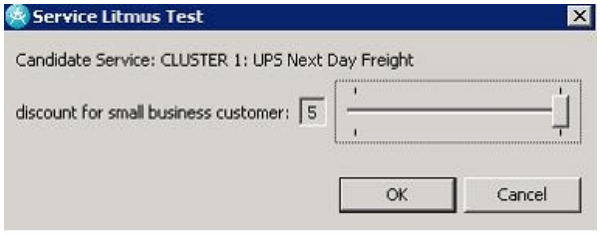


Fig. 15-c. Scoring the service of UPS Next Day Freight

Fig.15-d. Scoring the service of "UPS Second Day"

The scoring processes indicate that the service of "UPS Next Day Freight" tends to give more discounts to the target customer. The exposure decision module will output the "UPS Next Day Freight" as the best choice.

Fig.16. Selected service to be exposed

If there are more than one eligible service providers after comparing against the exposure threshold, ASLT toolkit allows the user to further filter out the unexpected services by launching the "Service Litmus Test Weighting" multiple times along with different exposure criteria. Each criterion, as well as its threshold, is used to filter out a set of service providers.

6 Conclusion

In this paper, we focus on addressing the scalability issue arising from designing the hybrid service asset management system. We propose a multi-level service hierarchy construction and management scheme that organizes services using pattern recognition algorithms that exploit the embedded services attributes. This research represents our effort in building comprehensive service asset management capability in the context of SOA solution design. There are several issues left to be resolved. In section 2, we state that the centralized management engine (BEDE) is connected with localized management console (LSMC) via service level agreements (SLAs). This connection plays a critical role in the lifecycle of the services asset management system. In specific, a systematic research needs to be conducted on various issues including formalizing, designing and monitoring the SLAs in the context of the interactions between BEDE and LSMC.

Acknowledgment

The authors wish to thank N. Zhou, Y.-M. Chee, A. Allam, A. Jalaldeen, K. Ponnalagu, R.R. Singdhgatta, A. Arsanjani, and F.Bernardini for their helpful comements and discussions. This work is supported by the project of SOMA-ME, a IBM's model-driven platform for SOA solution design.

References

[1] A. Arsanjani, S.Ghose, A. Allam, T. Abdollah, S. Ganapathy, K.Hoeely, "SOMA: A Method for Developing Service-Oriented Solutions," *IBM Sytems Journal*, vol. 47, no. 3, pp. 377-396, 2008.

[2] N. Apte and T. Mehta, *UDDI: Building Registry-Based Web Services Solutions*. Prentice Hall PTR, 2002.

[3] D. Ardagna and B. Pernici, "Global and Local QoS Constraints Guarantee in Web Service Selection," *Proc. Int'l Web Services (ICWS'05)*, pp.11-15, 2005.

[4] A.Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah, "S3: A Service-Oriented Reference Architecture," *IT Profesional*, vol. 8, no. 3, pp. 10-17, 2007.

[5] D.Balzarotti, C.Ghezzi, and M.Monga, "Supporting Cooperative Software Processes in a Decentralized and Nomadic World," *IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.36, no.6, pp.410-424, March. 2008.

[6] J.Basak and M.Gupta, "Active Evaluation and Ranking of Multiple-Attribute Items Using Feedforward Neural Networks," *IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.36, no.6, pp.1135-1145, November.2006.

[7] C.M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] K. Belhajjame, S.M. Embury, N. W. Paton, R. Stevens, and C.A. Goble, "Automatic Annotation of Web Services Based on Workflow Definitions," *ACM Trans. Web*, vol. 2, no. 2, pp. 11:1-11:34, 2008.

[9] Y.Chen, D.M.Kilgour, and K.W.Hipel, "Using a Benchmarking in Case-Based Multiple-Criteria Ranking," *IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.39, no.2, pp.358-368, March. 2009.

[10] S.X. Cheng, C.K. Chang, and L.-J. Zhang, "Stochastic Modeling Study for Competitive Web Services Market,"*Proc. IEEE International Conference on Web Services*, pp.960-967, 2007.

[11] S.X. Cheng, C.K. Chang, and L.-J. Zhang, "Modeling and Analysis of Performance Oriented and Revenue Based Admission Control," Proc. IEEE Congresss on Services, pp. 9- 16, 2007.

[12] T.-H. Cheng and C.-P. Wei, "A Clustering-Based Approach For Integrating Document-Category Hierarchies," IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans, vol.38, no.2, pp.410-424, March. 2008.

[13] G. Canfora, M. Penta, R. Esposito, and MM. K. Villani, "QoS-Aware Replanning of Composite Web Services," *Proc. Int'l Conf. Web Services(ICWS'05)*, pp. 121-129, 2005.

[14] C. Constantinopoulos, M. K. Titsias, and A. Likas, "Bayesian Feature and Model Selection for Gaussian Mixture Models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 1013-1018, 2006.

[15] A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, "Web Services on Demand: WSLA-Driven Automated Management," *IBM Systems Journal*, vol. 43, no. 1, pp. 136-158, 2004.

[16] Z. Dawy, B.Goebel, J.Hagenauer, C.Andreoli, T.Meitinger, and J.C.Mueller, "Gene Mapping and Marker Clustering Using Shannon's Mutual Information," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol.3, no.1, pp.47-56, 2006.

[17] P. D'haeseleer, S.D. Liang, and R. Somogyi,"Genetic Network Inference from Co-Expresssion Clustering to Reverse Engineering," *Bioinformatics*, vol.16, no.8, pp.707-726, 2000.

[18] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*. Willey-Interscience, 2000.

[19] F. Engel, "The Role of Service Level Agreement in the Internet Service Provider Industry," *International Journal of Network Management*, vol. 9, no. 5, pp. 299-301, 1999.

[20] W.J. Fang, L. Moreau, R. Ananthakrishnan, M. Wilde, and I. Foster,"Exposing UDDI Service Descriptions and Their Metadata Annotations as WS-Resources," *Proc. Int'l Conf. Grid Computing*, pp.128-135, 2006.

[21] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*. Wiley-Interscience, 2008.

[22] F. Harada, T. Ushio, and Y. Nakamoto, "Adaptive Resource Allocation Control for Fair QoS Management," *IEEE Trans. Computers*, vol. 56, no. 3, pp.344 – 357, 2007.

[23] Z. Y. He, X.F.Xu, and S.C.Deng, "K-ANMI, A Mutual Information Based Clustering Algorithm for Categorical Data," *Information Fusion*, vol.5, no.2, pp.223-233, 2008.

[24] S. Huang, Z .Cheng, Y. Yu, and W-Y Ma, "Multitype Features Coselection for Web Document Clustering," *IEEE Trans. Knowledge and Data Engineering*, vol. 18, no. 4, pp. 448-459, 2006.

[25] F. Jammes and H. Smit, "Service-Oriented Paradigms in Industrial Automation," *IEEE Trans. Industrial Informatics*, vol. 1, no. 1, pp.62-70, 2005.

[26] R.A. Johnson and D.W. Wichern, *Applied Multivariate Statistical Analysis (6th Edition)*. Prentice Hall, 2007.

[27] J.H.Liang, V.K.Vaishnavi, and A.Vandenberg, "Clustering of LDAP Directory Schemas to Facilitate Information Resources Interoperability Across Organizations," *IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.36, no.4, pp.631-642, July.2006.

[28] G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon,"Building Web Services for Scienfific Grid Applications," *IBM Journal of Research and Development*, vol. 50, no. 2/3, pp. 249-260, 2006.

[29] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services," *Journal of Network and System Management, Special Issue on E-Business Management*, vol. 11, no. 1, pp.57-81, 2003.

[30] P. Louridas, "Orchestrating Web Services with BPEL," *IEEE Software*, vol. 25, no. 2, pp. 85-87, 2008.

[31] F. Leymann, D. Roller, and M.-T. Schmidt, "Web Services and Business Process Management," *IBM Systems Journal*, vol. 41, no. 2, pp.198-211, 2002.

[32] H. Liu and L. Yu, "Toward Integrating Feature Selection Algorithms for Classification and Clustering," *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no.4, pp. 491-502, 2005.

[33] R.B. Mayerson, *Game Theory: Analysis of Conflict*, Havard University Press, 1997.

[34] R. Nock and F. Nielsen, "On Weighting Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no,. 8, pp. 1223-1235, 2006.

[35] T. Neubauer and C. Stummer, "Extending Business Process Management to Determine Efficient IT Investments," *Proc. ACM Symp. Applied Computing (SAC'07)*, pp. 1250-1256, 2007.

[36] Organization for the Advancement of Structural Information Statndards (OASIS), Universial Description Discovery and Integration (UDDI), July. 2002.

[37] H.Y. Paik, B.Benatallah, and F.Toumani, "Toward Self-Organizing Service Communities," *IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol.35, no.3, pp.408-419, May.2005.

[38] M. Perry and H. Kaminski, "SLA Negotiation System Design Based on Business Rules," *Proc. Int'l Conf. Services Computing (SCC'08)*, pp.609-612, 2008.

[39] H. Peng, F. Long, and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no.8, pp.1226-1238, 2005.

[40] A. Patrikainen and M. Meila, " Comparing Subspace Clusterings, " *IEEE Trans. Knowledge and Data Engineering*, vol. 18, no. 7, pp. 902-916, 2006.

[41] Rational Software Architect, IBM Corporation, http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html.

[42] W.N. Robinson, "Monitoring Web Service Requirements," *Proc. Conf. Requirements Engineering (RE'03)*, pp.65-74, 2003.

[43] J. Rosenberg ang D. Remy, *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. Sams, 2004.

[44] K. Sivashanmugam, K. Verma, and A. Sheth, "Discovery of Web Services in a Federated Registry Environment," Proc. Int'l Conf. Web Services (ICWS' 04), pp. 270-278, 2004.

[45] J.Wang, P.Neskovic and L.N.Cooper,"Improving Nearest

Neighbor Rule with a Simple Adaptive Distance Measure," *Pattern Recognition Letters*, vol.28, pp.207-213, 2007.

[46] *Websphere Integration Developer*, IBM Coporation, http://www-306.ibm.com/software/integration/wid/.

[47] H.Xiong, J.j. Wu and J.Chen, "K-Means Clustering Versus validation Measures: A Data-Distribution Perspective, IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 39. no.2, 2009.

[48] J. Yu, "General C-means Clustering Model," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1197-1211, 2005.

[49] L. Yu and H. Liu, "Efficient Feature Slection via Analysis of Relevance and Redundany," *The Journal of Machine Learning Research*, vol. 5, pp.1205-1224, 2004.

[50] L.A. Zadeh and C.A. Desoer, *Linear System Theory: The State Space Approach*, Dover Publications, 2008.

[51] L.-J. Zhang and B. Li, "Requirements Driven Dynamic Services Composition for Web Services and Grid Solutions," *Journal of Grid Computing*, vol.11, pp.121-140, 2001.

[52] L.-J. Zhang and Q. Zhou, Aggrate UDDI searches with Business Explorer for Web Services, IBM developerWorks, available at http://www.ibm.com/developerworks/webservices/library/ws-be4ws/, March 2002.

[53] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer and Tsinghua Universirty Press, 2007.

[54] L.-J. Zhang, A. Arsanjani, A. Allam, D. Lu and Y-M Chee, "Variation-Oriented Analysis for SOA Solution Design," *Proc. Int'l Conf. Servvices Computing (SCC'07)*, pp. 560-568, 2007.

[55] L.Z. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Cheng, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Engineering*, vol. 30, no. 5, pp. 311- 327, 2004.

[56] L.-J. Zhang, S.X. Cheng, Y.-M. Chee, A. Allam, and Q. Zhou, "Pattern Recognition Bsed Adaptive Categorization Technique and Solution for Services Selection," *Proc. The 2nd IEEE Asia-Pacific Service Computing Conference*, pp. 535-543, 2007.

[57] L.-J. Zhang, T. Chao, H. Chang, and J.-Y. Chung, "XML-Based Advanced UDDI Search Mechanism for B2B Integration," *Electronic Commerce Research*, vol. 3, no. 1-2, pp. 25-42, 2003.

[58] L.-J. Zhang, N. Zhou, Y.-M. Chee, A. Jalaldeen, K. Ponnalagu, R. R. Sindhgatta, A. Arsanjani, and F. Bernardini, "SOMA-ME: A Platform for the Model-Drven Design of SOA Solutions, " *IBM Systems Journal*, vol. 47, no. 3, pp. 397-413, 2008.

The Service-Oriented Solution Stack provides an architectural framework based on a range of industry best practices.

Ali Arsanjani, Liang-Jie Zhang, Michael Ellis, Abdul Allam, and Kishore Channabasavaiah

S3: A Service-Oriented Reference Architecture

For most businesses, a service-oriented architecture offers considerable flexibility in aligning IT functions and business processes and goals. An SOA decouples reusable functions, for example, and lets an organization externalize quality-of-service (QoS) variations in declarative specifications such as WS-Policy and related standards. As a flexible, extensible architectural framework, SOA reduces cost, increases revenue, and enables rapid application delivery and integration across organizations and siloed applications.

There's a challenging downside to SOA, however, in that it's significantly difficult to create an SOA solution. The architect must figure out how to produce a solution using a well-defined notation or how to organize the solution as an architectural framework with interconnected architectures and transformation capabilities. There is also the question of how to design for reusability and which tools will take the guesswork out of architecture validation and capacity planning.

To address these issues, we looked at projects from 2003 to 2006 to determine what an SOA architectural template, or *reference architecture*, would need. The result is the Service-Oriented Solution Stack (S3), which provides a detailed architectural definition of an SOA across nine layers that aim to reinforce business value. The underlying metamodel—the model for instantiating S3 for a given SOA solution—captures architectural building blocks—reusable functional elements that one or more components or products can realize—as well as the relations among these blocks and among layers, interaction patterns, options, and architectural decisions. The architect uses all these elements to create an SOA solution.

Because each layer represents different business value perspectives, the layers effectively separate concerns. The architect can easily create an SOA in concert with methods such as Service-Oriented Modeling and Architecture (SOMA; http://www-128.ibm.com/developerworks/webservices/library/ws-soa-designl/). Consequently, S3 not only enumerates the fundamental elements of an SOA solution, but also provides the ingredients and flexibility to model, architect, assemble, deploy, and manage that solution in a way that best fits a particular organization.

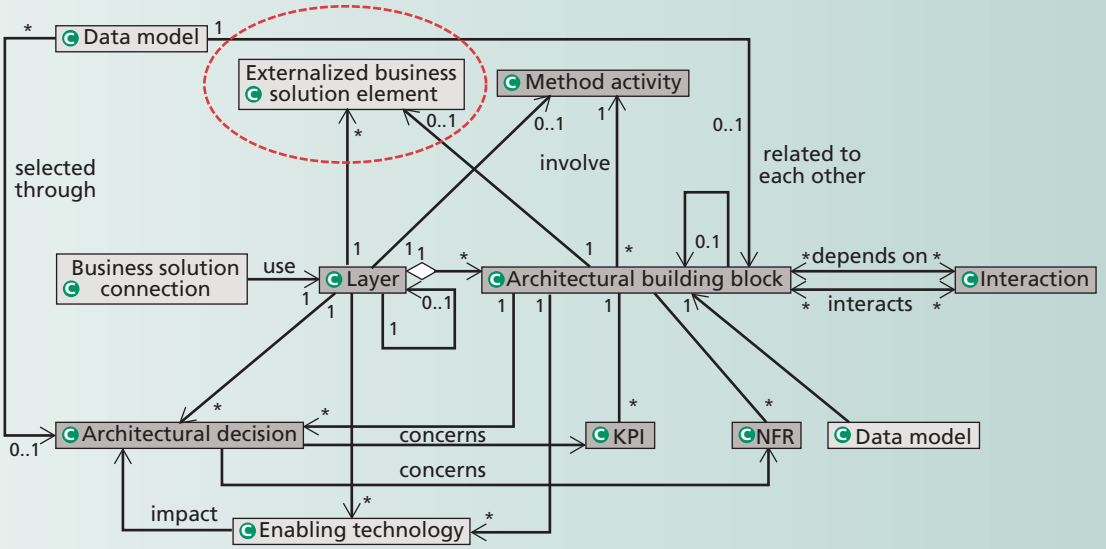
ADDRESSING THE LOGICAL

Each of S3's nine layers has a logical and physical aspect. The logical aspect includes all the architectural building blocks, design decisions, options, key performance indicators, and so on; the physical aspect covers the realization of each logical aspect using technology and products.

Our focus in this article is on the logical aspect, which addresses the question, If I build an SOA, what would it look like conceptually and what abstractions must be present? The metamodel in Figure 1 answers that question with the following elements: The dark gray boxes represent more static elements and the light gray ones the more

Published by the IEEE Computer Society
1520-9202/07/$25.00 © 2007 IEEE

Figure 1. A metamodel for instantiating S3 for a given SOA solution.



The block marked "layer" represents S3's nine layers. The darker gray blocks represent more static elements, while the lighter gray blocks represent more dynamic elements that vary according to the process or method used to populate S3.

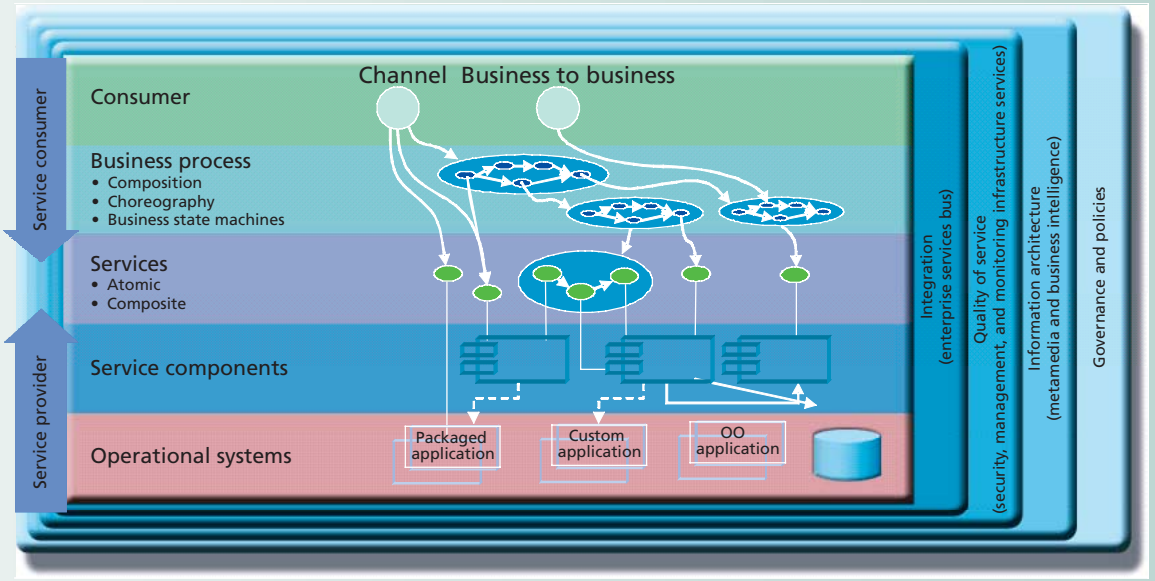
dynamic elements that would be attributed to a process or method used to populate the Service-Oriented Solution Stack, which is also known as the SOA Solution Stack. An example of such a method is the Service-oriented Modeling and Architecture (SOMA) method:

- *Layer.* An abstraction of the nine layers in S3; contains a set of components such as architectural building blocks, architectural decisions, and interactions among components and among layers.
- *Options.* A collection of options available in each layer that impacts other artifacts of a layer; the basis for architectural decisions within and between layers.
- *Architectural decision.* Derived from options, provides information on the configuration and use of architectural building blocks.
- *Method activity.* A collection of steps that show how architectural building blocks form a process in a layer.
- *Architectural building block.* Resides in a layer and contains attributes, dependencies, and constraints as well as relationships with other such blocks in the same or different layer.
- *Interaction pattern.* An abstraction of the various relationships among architectural building blocks, such as patterns and diagrams.
- *KPI.* A key performance indicator constraint on the architectural building blocks.
- *NFR.* A nonfunctional requirement constraint on the architectural building blocks.
- *Enabling technology.* A technical realization of the architectural building blocks in a specific layer.
- *Externalized business solution element.* A business service entity in a specific layer to be exposed to external consumers.
- *Business solution connection.* An adaptor for using external services.
- *Data model.* A model of the data content associated with architectural building blocks, including data exchange among layers and external services.

S3 assumes that a functional service requirement has two views. The provider view is the business and technical capability that a service must deliver to satisfy its consumers, while the consumer view is the business and technical capability that the service is expected to deliver in the context of that consumer alone. Figure 2 (next page) shows an SOA as a set of logical layers that are relatively independent, which lets an organization choose the degree of consumer-provider integration. Some solutions might use only a subset of the S3's nine layers; others might use all the layers. The degree to which an organization realizes the full S3 depends on the maturity of its service integration.

Figure 2 illustrates the multiple separations of concern in S3's nine layers. S3 does not assume that the provider and consumer are in one organization, although such collocation is common. Rather, S3 recognizes that business relationships are diverse: Some organizations might desire a high degree of consumer-provider integration; others,

Figure 2. Logical layers in S3.



The nine layers are relatively independent, which lets the organization choose the degree of consumer-provider integration. An SOA solution might exclude a business process layer, for example, and have the consumer and service layers interact directly. Services are, of course, part of both consumer and provider views. The lower layers (services, service components, and operational systems) are provider concerns, while the upper layers (services, business processes, and consumers) are consumer concerns.

not so much. Some organizations might be entirely consumers; others, entirely providers. S3 is flexible enough to accommodate any of these scenarios—from a tightly integrated consumer-provider relationship to one that is entirely decoupled.

THE LAYERS

In creating S3, we made several assumptions about the nine layers. We assume the existence of a set of service requirements that are both functional and nonfunctional and collectively establish the SOA's objective. Nonfunctional service aspects are security, availability, reliability, manageability, scalability, latency, and the like.

We also assume that a single layer or some combination of layers can fulfill any service requirement and that for each layer, the service requirements use a specific mechanism to influence that layer.

Finally, we assume that identifying service requirements and mapping them to the appropriate S3 layer is a critical part of developing an SOA.

S3's nine layers are operational systems, service component, services, business process, consumer, integration, QoS, information architecture, and governance and policies. There is no separate layer for business rules and policies. Rather, business rules cut across all layers: The business process and governance layers intersect in defining the rules and policies for the business process, and the input and output transformations from and to the consumer layer must abide by some business rules.

1. Operational systems

This layer includes all application assets running in an IT operating environment that supports business activities, whether custom, semicustom, or off the shelf. Because the layer consists of existing application software systems, implementing the SOA solution leverages existing IT investment. This in turn can lower the overall implementation cost and free up some of the overall budget for newer initiatives and the development of business-critical services. Software systems in the operational systems layer include

- existing monolithic custom applications, including Java2 Enterprise Edition (J2EE) and .Net applications;
- legacy applications and systems;
- existing transaction processing systems;
- existing databases; and
- existing package applications and solutions, including enterprise resource planning (ERP) and customer relationship management (CRM) packages, such as those from SAP or Oracle.

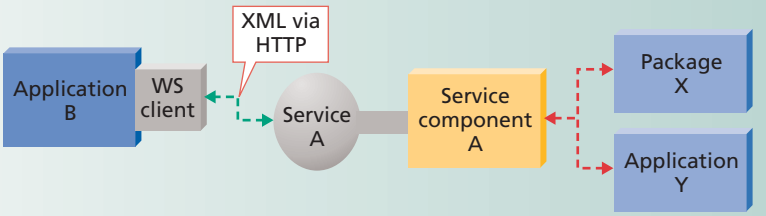
2. Service component

This layer contains software components, each of which are the realization of a service or operation on a service. Service components reflect both the functionality and QoS for each service they represent.

Each service component

- provides an enforcement point for ensuring QoS and service-level agreements,
- flexibly supports the composition and layering of IT services, and
- hides volatile implementation details from consumers.

Figure 3. The service component as a facade.



A provider has implemented Service Component A using a combination of behavior from third-party Package X and Application Y. Application B, the consumer, is coupled only to the description of the exposed service. Consequently, any later decision to replace third-party software will not affect Service Component A's consumers.

In effect, the service component layer guarantees that the IT implementation aligns with the service description. The detail-hiding characteristic is particularly powerful. In Figure 3, Service Component A acts as a service implementation facade by aggregating available system behavior and giving the provider an enforcement point for service compliance. The consumer must assume that the realized service is faithful to its published description (service compliance), and the providers must ensure that such compliance is achieved. The details of the realization, however, are of no consequence to Application B, the consumer.

At some later date, the provider organization might decide to replace Package X with Package M. Because the required modifications are encapsulated in the service component, such a decision does not affect any of its consumers.

Given SOA's recursive nature, service components can also be service consumers, a choice some organizations make to overcome integration challenges. Exposing internal services, such as those that support the infrastructure for business services, does not necessarily require the same analytic rigor as that for exposing a business service, which is traceable directly to business needs. An organization might have compelling IT-related reasons for using internal services, but they are generally not tied to any business process.

3. Services

The services layer consists of all the services defined within the SOA. In the broadest sense, services are what providers offer and what consumers or service requesters use. In S3, however, we define service as an abstract specification of one or more business-aligned IT functions (operations). The specification provides consumers with sufficient detail to invoke the business functions exposed by a service provider—ideally in a way that is platform independent.

Each service specification *must* include

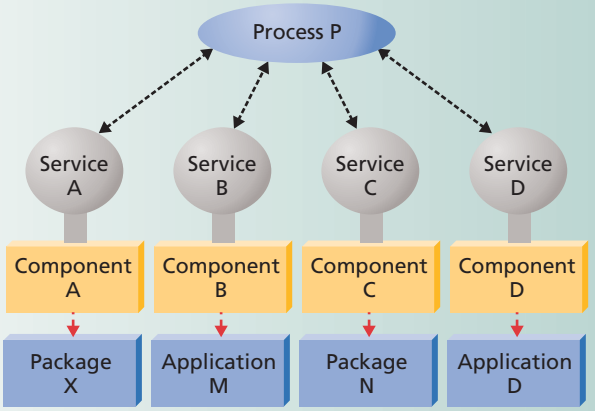
- operation signatures, such as those in the abstract stage of a Web Services Description Language description (but not necessarily in WSDL);
- service endpoint information (the network location to which invocation messages are sent);
- invocation protocol details; and
- service semantics, such as units of measure and business context.

Each service specification could also include service dependency information, such as shared state details; a policy document, SOA management descriptions, attachments that categorize or show service dependencies, and classification information.

Some services in this layer might be versions of other services in the set, which implies significant successor or predecessor relationships. Such relationships make it possible to discover, invoke, and choreograph services to create a composite service. Discovery and invocation occurs through well-defined service interfaces. The services layer provides a placeholder to mutually position an enterprise's service portfolio. Services in the service layer will externalize the interfaces of backend systems in the form of service descriptions, whether they are enterprise-scale, business-unit-specific, or project-specific components.

The exported service descriptions also contain the contracts that bind provider and consumer. The contract defines layers and their underlying building blocks according to the service identification activities. These activities in turn are defined through the use of domain decomposition, existing asset analysis, and goal-service modeling. IBM, for example, uses SOMA to model service identification activities (A. Arsanjani, "Service-Oriented Modeling and Architecture," IBM DeveloperWorks; http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/). Such activities represent the heart of the SOA value proposition—improved agility from decoupling business processes and IT. In some cases, the quality of these service

Figure 4. Orchestrating services.



An organization has created a business process, Process P, using Services A, B, C, and D from the services layer. Process P contains the logic for the sequence in which the services must be invoked and executed.

definitions can make or break an SOA, so an enterprise must select services carefully, using criteria such as the service litmus tests in SOMA's specification phase.

Another feature of the services layer is that service descriptions are independent of implementation and transport. Consequently, an organization can expose a service consistently across multiple customer-facing channels such as the Web, interactive voice response (IVR) systems, and the Seibel client (a system that customer service representatives commonly use). The integration layer (layer 6) handles the transformation of the channels' response.

4. Business process

In this layer, the organization assembles the services exposed in the services layer into composite services that are analogs for significant business processes. In the non-SOA world, these business processes are similar to custom applications. An example is the process of completing and submitting a loan application, which an organization typically performs through a custom application. An SOA in contrast would support that process by creating a composite service that choreographs the information flow among a set of services and human actors.

An SOA is particularly desirable in light of service reusability across disparate business processes. Organizations can assemble processes from a catalog of available service building blocks and reconfigure existing processes by shuffling the same blocks. The result is often large-scale software reuse and increased business agility.

By using technologies such as Web Services for Business Process Execution Language (WS-BPEL), organizations can assemble business processes just by modeling and manipulating a graphical depiction of process flow. Figure 4 shows how an organization can use tools that enable BPEL composition to implement a business process by aggregating services.

To enable interactions among services and business processes, the business process layer uses data- and control-flow techniques. The logical aspect of this layer covers all aspects of composition, collaboration, compliance, process library, process service, and invocation elements. The layer's physical aspect includes a runtime process engine, such as one based on WS-BPEL. Interaction can take place both within and across enterprises, and information exchange among participants (both users and business entities), resources, and processes takes a variety of forms. Exchanged information can also include messages that are not structured or transactional.

Business logic is the basis for modeling service flow as parallel tasks or sequential tasks based on rules, policies, and other business requirements. From the dataflow perspective, the business context and metadata support the aggregation of services within an enterprise to orchestrate or choreograph business processes. The business process layer also covers life-cycle management for such orchestration and choreography.

In a top-down approach to identifying services, business analysts define processes on the basis of customer requirements. To optimize the business process for better IT implementation, they break each process into activities that can be refactored as reusable services—activities that the organization can model, analyze, and optimize according to requirements, such as QoS, flow preference, price, time of delivery, and customer preferences. In a bottom-up approach, after creating a set of assets, the organization attempts to leverage them in a meaningful business context to satisfy customer requirements. By composing services guided by business requirements and composition rules, it can create reusable service assets, essentially enabling business processes on demand that address critical customer requirements.

As these descriptions imply, S3's business process layer plays a central coordinating role in connecting business-level requirements and IT-level solution components through collaboration with six layers: From the interaction perspective, the business process layer communicates with the consumer layer to communicate inputs and results with role players, such as the user, decision makers, and system administrator, through Web portals or business-to-business (B2B) programs. The integration layer transforms most of the business process's control-flow and dataflow messages, while the information architecture layer defines the message structures.

The key process indicators for each task or process can be defined in the QoS layer. The governance and policies layer guides the design of service aggregations. Finally, the business process layer must interact with the services represented and described in the services layer.

From the technical perspective, effective business process composition poses critical challenges to researchers and practitioners in Web services. Business processes are driven by business requirements, which typically tend to be informal, subjective, and difficult to quantify. Therefore, translating descriptive and subjective requirements into quantifiable, objective, and machine-readable formats is critical to effective process composition. Current Web services specifications generally lack the ability to define comprehensive relationships among business entities, services, and operations—relationships important to optimizing process composition. How to clearly specify search requirements to discover the most appropriate Web services candidates remains a challenge.

Another challenge is service collaboration. To satisfy business requirements, a typical business process generally requires the collaboration of multiple Web services. Thus, each service must satisfy not only its individual requirements, but must also coexist with other services to fit within the overall composed business process. This strongly implies the need to optimize the entire business process before its execution.

The business process layer addresses both these challenges, which further differentiates S3 from other conceptual reference architectures.

5. Consumer

The consumer layer handles interaction with the user or with other programs in the SOA ecosystem. Through it, an organization can deliver existing IT functions and data to applications or users according to specific user preferences. Consequently, an organization can quickly create the front-end of business processes and composite applications to respond to changes in the marketplace and enable channel-independent access to business processes and services exposed within a given SOA. This in turn allows the selection of rich client-user interfaces, which tend to use technologies that allow Web services to be invoked from the consumer layer. Such interfaces provide a framework that content providers can use to write consistent, interoperable Web services for portals.

S3 is particularly suitable for adopting proven front-end access mechanisms, such as portals, and open standards, such as Web Services for Remote Portlets (WSRP). Such mechanisms can decrease development and implementation cycle times and enable the reuse of prebuilt access mechanisms, thereby reducing complexity and maintenance costs.

The use of WSRP, in particular—an open standard that leverages Web services at the application interface or presentation level—promotes a single unified view of knowledge presentation and a consistent access mechanism for exposed business processes and applications that integrates with other foundational services such as security (single sign-on, for example) and trust. It also allows for the plug and play of content sources, such as portlets, with portals and other aggregating Web applications.

WSRP functions are exceptionally valuable to organizations that must provide presentation-oriented Web services suitable for use in a portal or those that prefer to adopt an open standard for making business processes and services available in a portal. Using WSRP, organizations can host content in the environment most suitable for its execution, yet keep it accessible to content aggregators. In other words, content producers can maintain control over the code that formats the content's presentation. By reducing the cost for aggregators to access their content, WSRP increases the rate at which an organization can integrate content sources into pages for user access.

WSRP and AJAX are just two of many SOA solutions that decouple the user interface from the consumer layer components.

Another technology gaining importance in the consumer layer is AJAX, short for Asynchronous JavaScript and XML (http://en.wikipedia.org/wiki/ajax). AJAX enables the exchange of XML contents over HTTP without the need for Web browser refreshing and provides a richer, more responsive user interface. Organizations can combine it with WSRP to enhance the consumer layer's user interaction capabilities.

WSRP and AJAX are just two of many SOA solutions that decouple the user interface from the consumer layer components: Others include the Web 2.0 style of mashup technologies, portlets, and B2B.

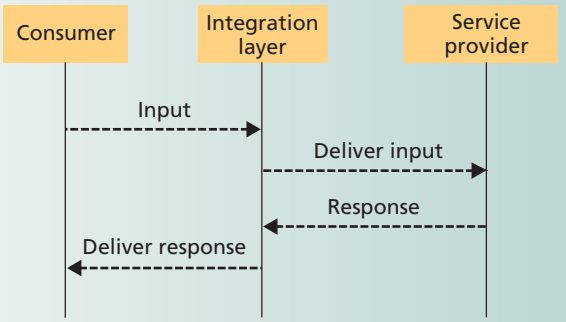
6. Integration

This layer integrates primarily layers 2 through 4, making it crucial to an SOA. The layer's integration capabilities let an organization mediate, route, and transport service requests from the service requester to the correct service provider. These capabilities include but are not limited to those found in an enterprise service bus (ESB):

- message transformation often associated with point-to-point tightly coupled endpoint integration,
- intelligent routing (high-availability, content-based, and so on), and
- protocol transformation.

An SOA often places requirements on the integration layer infrastructure to provide communication, invocation, and QoS enforcement between adjacent layers. This is where binding (late or otherwise) of services occurs for process execution, for example. Binding is what lets an organization expose a service consistently across multiple customer-fac-

Figure 5. Interactions in the integration layer.



Because a service consumer interacts with the service provider through this layer, an organization can expose the service's specification through a support mechanism such as an enterprise service bus.

ing channels such as the Web, IVR, and a Seibel client. XSLT functions supported through ESB transformation transform the response to HTML (for Web), voice XML (for IVR), and XML string (for a Siebel client).

As Figure 5 shows, the integration layer provides a level of indirection between consumer and provider, essentially decoupling the two. This provides enough flexibility for organizations to integrate disparate systems into new solutions.

7. Quality of service

Inherent in an SOA are characteristics that exacerbate existing QoS concerns in computer systems: increased virtualization, loose coupling, widespread use of XML, the composition of federated services, heterogeneous computing infrastructures, decentralized service-level agreements, the need to aggregate IT QoS metrics to produce business metrics, and so on. These characteristics create complications for QoS that clearly require attention in any SOA.

The QoS layer lets an SOA signal noncompliance with service qualities in each SOA layer. Thus, the SOA can capture (in an operational sense), monitor, log, and signal noncompliance with nonfunctional requirements that relate to the service qualities. In some cases (such as security), the QoS layer can actually realize such nonfunctional requirements. In a sense, this layer observes the other layers and emits signals or events when it detects noncompliance, or preferably when it anticipates noncompliance.

The QoS layer establishes issues related to nonfunctional requirements as a primary concern of SOA and provides a focal point for dealing with them. It provides the means of ensuring that an SOA meets its requirements with respect to reliability, availability, manageability, scalability, and security. Finally, it enhances the SOA's business value by letting organizations monitor the business processes in the SOA according to the key business process indicators they influence.

8. Information architecture

This layer ensures that an organization includes key considerations affecting data and information architectures, which an organization can also use to create business intelligence through data marts and warehouses. The layer includes stored metadata content.

For industry-specific SOA solutions, this layer captures the reference points for all the cross-industry and industry-specific data structure, XML-based metadata architectures (such as XML schema), and protocols of exchanging business data. It also covers the enabling frameworks of discovery, data mining, and analytic data modeling.

9. Governance and policies

The governance and policies layer covers all aspects of managing the business operations' life cycle. This layer includes all policies from manual governance to WS-Policy (where the services layer and governance and policies layer intersect). It provides guidance and policies for managing service-level agreements, including capacity, performance, security, and monitoring. As such, the governance and policies layer is applicable to all other S3 layers. From a QoS and performance metrics perspective, for example, it is tightly connected to the QoS layer. The layer's governance framework includes service-level agreements based on QoS and key process indicators, a set of capacity-planning and performance-management policies to design and tune SOA solutions, and security-enabling guidelines for composite applications.

The governance and policies layer can accelerate SOA solution planning and design and increase the sales opportunities for managing customers' existing solutions.

CREATING SOLUTIONS

Organizations design and implement SOA solutions by leveraging existing techniques and technologies, many of which have an associated set of best practices not specifically related to the SOA. For example, writing robust J2EE applications and components is an important part of building SOA solutions, although we have not mentioned this. Our focus for the most part is on what is critical to building effective SOAs.

The S3 applies to various types of practitioners such as enterprise and solution architects. Architects can use S3 as a checklist of layers, architectural building blocks and their relations in each layer, available options, and decisions that must be made at each layer. The layers provide a starting point for the separation of concerns needed to build an SOA.

A recurring theme in SOA projects is how to apply an SOA to address areas of increasing scope: a single project, a line of business, a few lines of business-sharing services,

enterprise scale, supply-chain (value-net), and a larger SOA ecosystem (multiple enterprises). In each case, organizations tend to apply SOA principles in the same way, which has led to the term *fractal use*.

When applying S3, SOA solution creators will typically need to create the same layers for each level of granularity. Thus, the enterprise architecture might use S3 as an SOA solution template customized or instantiated for each business or product line, depending on how the organization is structured. To be part of an SOA ecosystem, on the other hand, an organization would use S3 to facilitate integration and collaboration. Thus standardization would benefit companies at the architectural level just as it benefits them at the level of data interchange through XML and XML schema.

TOWARD VENDOR-NEUTRAL ARCHITECTURE

S3's overarching goal is to streamline the process of modeling and documenting architectural layers, building blocks, options, product mappings, and all the architectural and design decisions that are part of creating an SOA. By providing a template and guidelines for architectural design, S3 answers questions that have long plagued SOA architects: What layers must I look at, what building blocks must I consider, and what decisions must I make when choosing a set of architectural building blocks within a layer?

Because it can objectively identify SOA infrastructure requirements, S3 is also useful in designing vendor-neutral solutions. Architects can focus on the parts of an SOA solution that are important in the context of the problem at hand and map the required capabilities onto available products with the required capabilities. This contrasts sharply to the typical approach, which is to try to reverse-engineer an SOA solution from the capability of a particular vendor's products. It is far more efficient to select the best fit from a mix of vendors—all of which offer the same architectural building block. In that sense, S3 can deliver SOA business services within the same implementation framework, varying only the capabilities required. ■

Acknowledgments

We appreciate the input and advice from colleagues Raj Cherchatil, Arnauld Desprets, Donald Ferguson, Rolando Franco, Biffle French, George Galambos, Kerrie Holley, Joe Hardman, Rao Kormili, Min Luo, Emily Plachy, Siddarth Purohit, Robert Rafuse, Rachel Reinitz, Rick Robinson, Raghu Varadan, Dan Wolfson, Olaf Zimmerman, Jamshid Vayghan, Petra Kopp, Jia Zhang, and other members of the IBM worldwide SOA practitioner community.

Ali Arsanjani *is an IBM Distinguished Engineer and chief architect of the SOA and Web Services Center of Excellence at IBM Global Business Services. He also leads the IBM SOA Community of Practice, which consists of more than 5,600 practitioners. Contact him at arsanjan@us.ibm.com.*

Liang-Jie Zhang *is a research staff member and lead of SOA Services Research at IBM T.J. Watson Research Center and worldwide lead of IEEE Services Computing Community. Contact him at zhanglj@us.ibm.com.*

Michael Ellis *is a solution architect at IBM Software Services for WebSphere. Contact him at msellis@ca.ibm.com.*

Abdul Allam *is a certified IT architect at IBM Business Consulting Services. Contact him at allam@us.ibm.com.*

Kishore Channabasavaiah *is a chief architect in the SOA and Web Service Center of Excellence at IBM Business Consulting Services. Contact him at kishorec@us.ibm.com.*

Emerging patterns in the use of XML for information modeling in vertical industries

S. Hinkelman
D. Buddenbaum
L.-J. Zhang

Extensible Markup Language (XML) has emerged as the predominant non-binary information format. The impact of XML has been most strongly felt in information exchange environments and information modeling. This paper focuses on a set of innovative patterns that has emerged in the use of XML for information modeling and business content design in the health-care, travel, insurance, and other industries. We provide historical perspectives on this development and characterize XML's current state in relation to Web Services.

INTRODUCTION

The use of XML (Extensible Markup Language) for information modeling within vertical industries has taken many diverse forms. Some, but not all, of these forms have been influenced by the emerging service-oriented architecture (SOA) XML infrastructures. Despite the diversity of approaches taken by industry-level consortiums working with XML, there is a great deal of commonality, as exemplified by four basic patterns for XML business content design which have recently emerged within vertical industry consortiums. These patterns are (1) Business Content Envelope, (2) Web-Services-Based Infrastructure, (3) Wrapped Content, and (4) Top-Down Modeling. This set of patterns, though limited, provides a framework that can aid Web Services adoption efforts by industry standards organizations.

In this paper, we begin with a review of the history of the development of a selection of XML standards. Next, we focus on the emergence of the aforementioned industry-level patterns in XML business content design and describe these patterns in detail. We then describe the associated effects and implication of mappings (i.e., "bindings") of these patterns to a Web Services infrastructure.

DEVELOPMENT OF XML STANDARDS

XML was originally designed for large-scale electronic publishing applications but has grown to handle the exchange of information in a variety of contexts. The fundamental standards activity for XML was conducted by the World Wide Web Consortium[1] (W3C**), beginning with the Extensible Markup Language[2] (XML) Version 1.0 W3C

Recommendation in February 1998. W3C continues to coordinate XML standards activity with working groups in such areas as XML Query, XML Schema, XML Core, and XML Processing Model.

Industry consortiums developing standards for XML information exchange have emerged over the last several years due to the explosive growth of XML and its ability to define structured vocabularies. Some of these organizations and their efforts are described in this paper. A number of these organizations have explicitly defined themselves as using today's commercial Web Services infrastructure for information exchange.

The Open Application Group, Incorporated[3] (OAGi) manages the Open Application Group Integration Specifications (OAGIS**) standard, which defines a business information envelope along with a set of business-information content types. The Association for Cooperative Operations Research and Development[4] (ACORD), the leader in global insurance standards, has developed messaging standards for Life and Annuity (L&A), Property and Casualty (P&C), and reinsurance products for the insurance industry. Like OAGi, it is a member-driven organization whose efforts began before Web Services were defined and widely accepted, and its mission includes staying current with infrastructure technologies such as Web Services. Remaining infrastructure-neutral like OAGi, ACORD has developed core specifications and complementary Framework Implementation Guides. As stated in the *ACORD Messaging Service XML Specification and SOAP Implementation Guide*, "The design of the ACORD core payload standards will not prohibit or intentionally favor use of any framework standards that are specified by cross-industry bodies. On the reverse, ACORD Framework Implementation Guides can be viewed as cross-industry standard profiles to support the insurance business processes in the most adequate way."[5] Like OAGi's approach, the ACORD approach is not to design Web services directly as a dependency, but to map onto Web Services capabilities. The OpenTravel Alliance** (OTA) serves a similar function for the travel industry. The standardization activities of OAGi, ACORD, and OTA are described in detail in this paper, along with those of some other companies.

As is the case in many fields, the nature of the organizations managing the development of standards has a great impact on the type and qualities of standards that are produced. In the following subsections, we list some of the organizational characteristics relevant in this context for XML standards.

Legacy-based vs "green field" organizations

Some organizations have a long history of standards work in information exchange, possibly going back to the days of Electronic Data Interchange (EDI). Their activity reflects this history, and tremendous effort tends to be spent on managing transitions between technical implementations, even to the degree of ensuring some level of backward and forward compatibility. These efforts often result in a suboptimal implementation. Other organizations lead "green field" standards efforts (i.e., those without a history), enjoying much more freedom to adopt current techniques, based on the "best of breed" thinking at the time that the development is taking place. For example, a "session" construct may be a necessary element for supporting reliable or sequenced messaging. A green field approach would typically rely on a recent horizontal (i.e., cross-industry) standard such as Web Services Reliable Messaging[6] (WS-RM), to provide this support; a more seasoned standard would probably use a legacy mechanism modeled within its architecture. In the latter case, for reasons of continuity (because approaches may be short-lived), ease of adoption, and consistency with production applications, the legacy approach may be perpetuated at the expense of, or in addition to, the application of a recently emerged standard, such as WS-RM, that provides the same function.

Comprehensive vs streamlined organizations

Some organizations mandate the development of solutions across the entire "eco-system" of implementations. This results in a more complete view of the scope of the problem, with generally wider acceptance, but with an increased burden of consensus building and use of existing production standards. This also results in a potentially more conservative approach to cross-industry standards adoption. Other efforts are managed by streamlined consortiums of like-minded organizations looking to accelerate development and adoption of a particular standard within a smaller scope of use. In this case, the effort is associated with solving a key aspect of a problem which the consortium has deep knowledge of and wishes to see addressed.

Information exchange vs service optimization focus

The organization's charter puts limits on the scope of the issues that the organization can and will address. Some standards organizations are developed to solve information sharing problems. In this case, the modeled content can be more document-oriented in an attempt to synchronize the data held by the various members of the organization, such as a standard for sharing customer data. Other standards organizations are concerned with service optimization. In this case, the modeled content is designed to efficiently provide a service, such as payments over a banking network. Still other standards organizations are more concerned about process optimization and take a more service-oriented approach aimed at bringing a process through various states to its conclusion, for example, from inception through completion of a purchase order.

Structure vs process focus

The maturity of the standards produced by an industry standards organization can be based on the degree to which it standardizes not only structure and syntax but also processes. Standardized processes provide a standardized context for the use of message formats derived from the structure and syntax of the business information model. This provides impetus for the standardized design of functions such as security, reliability, and routing. Specifying information interaction patterns among relevant technologies increases the value of the business content specification by facilitating adoption. Industry standards organizations wish to provide bindings to relevant technologies for this reason. Dominant business players tend to simplify interaction requirements, limiting them to the set of patterns and technologies that meet their business needs; industries with a more heterogeneous set of partners usually have a more complex set of required interaction patterns.

EMERGING BUSINESS CONTENT DESIGN PATTERNS

Clearly, there is no "one size fits all" approach for best practice implementations adopting Web Services in all of the vertical-industry standards organizations. Instead, there are many diverse approaches. While these divergences are significant, a basic set of patterns has emerged in recent years. This set of patterns is useful in understanding and aiding Web Services adoption efforts by industry standards organizations. The patterns simplify understanding the impact of each approach and help identify requirements that can increase the rate and sophistication of Web Services adoption.

In the following subsections, we explore these basic patterns for business content design: (1) Business Content Envelope, (2) Web Services-Based Infrastructure, (3) Wrapped Content, and (4) Top-Down Modeling. We discuss the differences between these patterns and describe some implementations that use them.

Business Content Envelope pattern

In order to endure and have a high level of insulation from change in underlying infrastructure technology, an industry-level standard is required to manage (at least) the basic specification of interaction and process indicators in a way that is consistent with, and has some level of integration with, the business content itself. A comprehensive pattern integrating business information with interaction and process indicators is referred to as a "Business Content Envelope" pattern. (An XML envelope is an XML document type that is defined to be a holder for other arbitrary XML data.) While not new, this pattern has proven to be successful and is currently used within various organizations. This pattern is based on the fundamental principles of abstraction of infrastructure and minimizing dependencies on any given information-exchange infrastructure.

The OAGIS standard uses this pattern and abides by some foundational infrastructure-neutral principles. OAGIS defines a Business Object Document (BOD), a comprehensive Business Content Envelope, along with business content. In the OAGIS envelope architecture the business content, referred to as a *noun*, is associated with multiple actions, which are referred to as *verbs*. Extensibility is also supported as a core part of the architecture. the OAGIS Business Content Envelope is by principle and by design independent of any specific lower-level protocol or transport since its inception. ACORD has also developed a Business Content Envelope.

In the following subsections, we describe in detail the approaches used by ACORD and OAGi in developing their Business Content Envelope patterns.

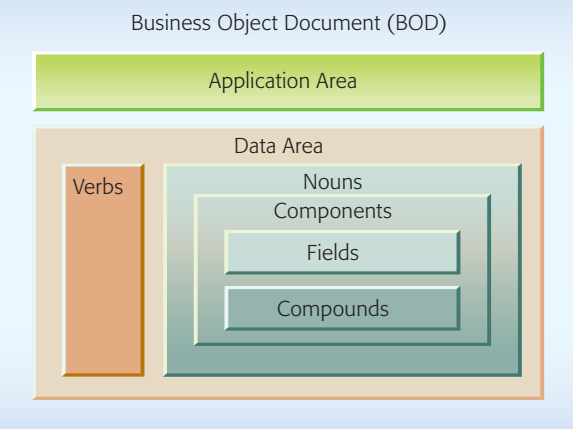


Figure 1
OAGIS BOD architecture

The OAGIS envelope

Figure 1 shows the overall architecture of the OAGIS envelope.[7] OAGIS schemas specify a naming convention for BODs, consisting of verbs and nouns such as `ProcessPurchaseOrder`. The outer layers of the BOD envelope identify the intentionality (i.e., what the message is intended to do, a verb), business content (a noun), version identifier of the document, release number of OAGIS, and a test/production flag, for example:

```
<ProcessPurchaseOrder...versionID="..."
  releaseID="..."
  systemEnvironmentCode="Production">.
```

An Application Area contains application-specific information common to all BODs, such as:

```
<ApplicationArea>
   <Sender>
     <LogicalID> ... </LogicalID>
     <ComponentID> ... </ComponentID>
     <TaskID> ... </TaskID>
     <ReferenceID> ... </ReferenceID>
     <ConfirmationCode> ...
     </ConfirmationCode>
     <AuthorizationID> ... </AuthorizationID>
   </Sender>
   <CreationDateTime> ... </CreationDateTime>
   <Signature ... />
   <BODID ... />
   <UserArea ... />
</ApplicationArea>
```

A Data Area carries the business content, which is the information that is specific to each BOD envelope, as shown in *Figure 2*. OAGIS further defines the business content in a hierarchy of elements, called components, fields, and so forth.

The Acord envelope

ACORD designed a messaging service, the Acord Messaging Service Version 1.2, which wraps each of the insurance industry XML standards (P&C, L&A, and reinsurance) based on a set of horizontal standards as shown in *Figure 3*. The purpose of Acord Messaging Service Version 1.2 is to support the transport, routing, content, and security requirements of the L&A, P&C, and reinsurance standard specifications. ACORD's stated position is to create similar implementation guides as technologies evolve. The approach taken for Web Services enablement is to create an Acord Messaging Service Version 1.2 that provides an envelope which supports specific requirements from multiple standard specifications in a technology-neutral way, including message management (requiring a unique identifier, specific message type, status, and message signature for nonrepudiation), routing (requiring sender/receiver, time stamp, and intended application), packaging, and security.

ACORD is currently developing the next version of the Acord Messaging Service Version 1.2 specification, which will be called the ACORD Web Services Profile. The profile will still be based on the Business Content Envelope pattern, but will differ from today's specification by the design of the SOAP and WSDL bindings. The goal is to align more closely with SOA principles and enable business services to be exposed by WSDL. This will be enabled by profiling the latest development of Web Service Standards (SOAP 1.2, WSDL 2.0, WS-addressing and WS-Reliable Messaging) and promoting harmonized service wrapper components within the business payloads in each of the three industry standards.

As an example, Figure 3 contains excerpts from the Acord Messaging Service Version 1.2 Business Content Envelope pattern of an ACORD message within a Simple Object Access Protocol (SOAP) body.[8]

Web Services-Based Infrastructure pattern

This pattern includes those industry-level patterns whose infrastructure is based exclusively on Web Services. Although consortiums following the Web Services-Based Infrastructure pattern use Web

```
<DataArea>
  <Process acknowledgeCode="Always">
    <ActionCriteria>
      <ActionExpression expressionLanguage="token" actionCode="Add">token</ActionExpression>
      <ChangeStatus>
       ...
      <EffectiveDateTime>...</EffectiveDateTime>
      <ReasonCode>...</ReasonCode>
       ...
        <StateChange>
           <FromStateCode>...</FromStateCode>
           <ToStateCode>...</ToStateCode>
           <ChangeDateTime>...</ChangeDateTime>
           <UserArea/>
        </StateChange>
      </ChangeStatus>
    </ActionCriteria>
  </Process>
  <PurchaseOrder>...</PurchaseOrder>
</DataArea>
```

Figure 2
Example of a data area

Services exclusively for their infrastructure, the business content design developed by these consortiums is typically developed without concern for the low-level infrastructure details of Web Services.

Unlike the Business Content Envelope pattern, this pattern defines "usage-context-free" instance documents (i.e., those containing no verbs, interaction indicators, process indicators, etc.), which can be referenced under several general operations that are defined in the infrastructure. This is in sharp contrast to a Business Content Envelope pattern, which contains much more than the business content. This content design pattern requires conforming instance documents to have a single specific root element.

The architecture of MedBiquitous.org,[9] a distinguished medical professional organization, was defined by reference to the Web Services-Based Infrastructure pattern. MedBiquitous.org explicitly and solely relies on the Web Services infrastructure to provide all interaction and process specifications for exchanging professional business content. A natural and complementary content design authoring pattern that fits well with this pattern, known in industry by some as the "venetian blind schema" pattern, is shown in the following example:

```
<xs:schema...
  <xs:complexType name="InternalElementType">
    <xs:sequence>
      <xs:element name="A"/>
    </xs:sequence>
   </xs:complexType>
   <xs:element name="RootElement"
                  type="RootElementType"/>
  <xs:complexType name="RootElementType">
    <xs:sequence>
      <xs:element name="InternalElement"
                   type="InternalElementType"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Binding conventions typically use a wrapped model for business content in which the business content schemas are wrapped with another schema that defines services operations. Within MedBiquitous .org, a wrapped Doc/Lit message style (i.e., one in which information is exchanged in a "raw" form, without encoding or other alteration) is used within the infrastructure layers specific to Web Services. The business content payloads are wrapped with request/response operation wrapper elements, which are defined in a separate schema. No first-class business content (i.e., actual business information, as opposed to associated information or metadata) is defined as part of these wrappers.[10] This marks the initial separation point between business content operations and those that are required for interactions by means of the Web

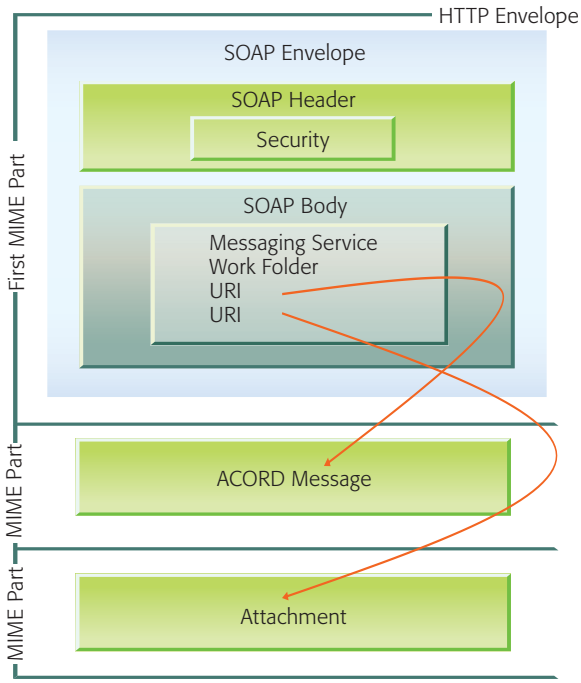


Figure 3
ACORD Messaging Service Version 1.2 architecture

Services infrastructure. The wrapper schemas, one per Web service, consist of the request and response elements for all operations defined within the service.

WSDL files import this schema. WSDL (e.g., the `<types>` structure) is *not* used directly for business content, in order to keep all business type information independent of the Web Services interface specification documents. This separation helps facilitate business payload development by shielding the domain experts from the details of the underlying Web Services technology. In the case of MedBiquitous.org, the actual payload schemas are indirectly imported into WSDL files through the wrapper schemas within the WSDL `<types>` structure, as in the following example:

```
<types>
   <xs:schema...
      (include declaration of the namespace wrapper)
      <xs:include schemaLocation=
         (location of the wrapper schema) />
   </xs:schema>
</types>
```

Business content is not authored within the WSDL files.

The imported wrappers then are used to define the request and response message parts. Naming conventions using WSDL operation names determine the message names, as shown in the following example:

```
<message name="Operation1InputMessage">
   <part name="Wrapper" element="operation1"/>
</message>
<message name="Operation1OutputMessage">
   <part name="Wrapper" element="operation1Response"/>
</message>
```

This design uses an approach in which generic interface operations are used across the entire multiusage payload. This shields the interface from change over time, but necessitates constraint checking within the business logic.

In the Web Service-Based Infrastructure pattern, the intentionality of the message is completely determined by the choice of WSDL operation names, unlike the Business Content Envelope pattern, where intentionality indicators, such as verbs or actions, typically exist as design elements integrated with the business content architecture. This is not to say that the business content design is completely defined without any idea or context of how the information is going to be used, but a Web Services-Based Infrastructure pattern formally makes the intentionality of the messages visible only at the Web Services infrastructure layer.

A complementary aspect to the usage-context-free business information modeling of this pattern lies in how it specifies infrastructure operation granularity. In order to avoid fragile Web service interfaces, general operations on service interfaces are used rather than highly specific operations. An example of this would be the use of the well-known Create/Read/Update/Delete (CRUD) operations rather than operations highly specific to the business content such as `CreateCardiologist`. Another complementary aspect is payload or content design authoring specifying a highly type-based document with a single root element, as discussed previously. The root element can be referenced by several general operations, as seen here:

```
<operation name="Create">
   <input message="Operation1InputMessage"/>
   <output message="Operation1OutputMessage"/>
```

```
  <fault...
</operation>
```

Like almost all industry groups addressing Web Services, interoperability is a key concern[11] for MedBiquitous.org. The WS-I basic profile provides a foundation for interoperability guidelines, but part of this profile is a clarification of Web Services specifications targeted for Web Services infrastructure developers and is not directly relevant to vertical-industry standards organizations. In MedBiquitous.org, areas such as common faults for all services operations are defined to supplement the basic profile in the context of the Web Services infrastructure and increase interoperability.

A ramification of general purpose operations typical of the Web Services-Based Infrastructure pattern as described here is that Web Services faults that are specific to the business content are typically not defined—hence, the need for general error definitions, such as those defined in MedBiquitous.org. One such error is a 'business rule' fault to accommodate an error associated with the business content, which is specified for every Web Services operation, and can be specified with the syntax `<fault name="BusinessRuleFault" message=../>`.

From an overall organizational and architectural view, the Web Services-Based Infrastructure pattern is arguably more efficient than the Business Content Envelope pattern, positioning an organization to take full advantage of the present and future functions of the Web Services infrastructure. Relying solely on the Web Services infrastructure eliminates concerns about infrastructure abstractions in higher-level designs. However, defining the infrastructure in a concrete way specific to a given infrastructure technology like Web Services means that when an alternative infrastructure emerges, significant wholesale replacement must be defined at the lower levels as there are no integrated interaction abstractions and process indications at the upper layers.

Wrapped Content pattern

The Wrapped Content pattern, similar to the comprehensive Business Content Envelope pattern, allows an organization to remain independent of a given message exchange infrastructure; it also requires some level of definition for interaction and process indication. The Wrapped Content pattern is limited and is based on wrapping primary business content with a single style of interaction such as request/response. There is no overall design of sections to contain information for specific purposes. In contrast, a Business Content Envelope pattern is robust and takes on many abstractions including processing actions (verbs), acknowledgements and confirmations, and an overall compartmentalized extensible stucture for containment of different information used for different purposes.

Some industry organizations, in addition to defining reusable business content, define a common set of reusable process indicators that are integrated with interaction exchange structures used as wrappers for the industry-specific business content. These process indicators are similar to the metadata defined in full Business Content Envelope patterns. In its most basic form, the business information is contained within an exchange structure such as a message request and a message response. ACORD, within its overall design and methodology, defines request/response interaction exchange structures.

The OpenTravel Alliance[12] (OTA), a well-respected pioneering organization for the travel industry, also uses this design approach. OTA has standardized a set of common attributes and indicators that may appear on the request/response interaction-exchange-structure root element for all OTA message payloads.[13]

OpenTravel Alliance and the Wrapped Content pattern

OTA predates current distributed infrastructures such as Web Services infrastructures. To illustrate the use of process indicators for this pattern, we use examples from OTA's "common types" schema and define partial example instances. The travel industry specifies many industry-specific reusable types within this schema, such as `LoyalLevel` and `HotelReference`, by using process indicators. OTA provides several indicators within its specifications to accommodate usages that are current in this industry.

An indicator may be specified when a requesting host indicates that the receiving host should include an 'echo token' of the same value in the response. Process indicators may be used to indicate the processing model (test or production) of the

```
<OTA_AirAvailRQ
     xmlns=http://www.opentravel.org/OTA/2003/05
     xmlns:xsi=
          "http://www.w3.org/2001/XMLSchema-instance"
     EchoToken="12345"
     TimeStamp="2003-07-17T09:30:47-05:00"
     Target="Production"
     Version="2.001"
     SequenceNmbr="1"
     PrimaryLangID="en-us"
     MaxResponses="10"
     DirectFlightsOnly="false"
     TransactionStatusCode="Continuation"
     TransactionIdentifier="224">
       <POS>
         <Source AgentSine="BSIA1234PM"
          PseudoCityCode="2U8"
          ISOCountry="US"
          ISOCurrency="USD">
          <RequestorID URL=
          "http://www.//provider1.org" Type="5" ID="123"/>
         </Source>
       </POS>
       <OriginDestinationInformation>
         <DepartureDateTime>2003-08-13</DepartureDateTime>
         <OriginLocation LocationCode="LHR"/>
         <DestinationLocation LocationCode="LAX"/>
       </OriginDestinationInformation>
       <TravelPreferences...
</OTA_AirAvailRQ>
```

Figure 4
Example of OTA Air Availability request instance

receiving node. The version of the message may also be indicated.

A unique identifier may indicate that all messages sent in a set of request and response messages are part of a single ongoing transaction, and a message sequence number can be used to identify the number of the transaction as assigned by the sending system. This numbering allows an application to process messages in a certain order or to request a resynchronization of messages in the event that a system has been offline and needs to retrieve messages that were missed.

A transaction status code may be defined to indicate where a specific message lies within a sequence of messages. The code may take the values `Start`, `End`, `Rollback`, `InSeries`, and `Continuation`. A process indicator may be defined to indicate the desired version of the payload response message. This requirement for specifying one of several non-error responses may provide a challenge when mappings to Web Services interfaces and operations are defined.

OTA's 2005A Air Availability schemas provide a comprehensive view of business content wrapped within an interaction exchange structure along with associated process indicators. These schemas specify the availability of flights between a pair of cities on a specific date for a specific type and number of passengers, as in ***Figure 4***. Many of the previous indicators are exemplified here within this Wrapped Content message.

OTA has not published formal binding specifications for a Web Services infrastructure. At present, only information for the June 2005 ebXML Message Service specification[14] (ebMS) is documented. Definition of the mapping to alternative infrastructure technologies is currently under development. However, OTA's robust interaction Wrapped Content pattern along with its process indicators would provide a comprehensive source for successful mapping to the evolving Web Services infrastructure with its increasing transactional and state-full capabilities, considering OTA's already defined transaction status codes and identifiers.

ACORD and Wrapped Content

ACORD XML for L&A products is based on the ACORD Life Data Model and provides a robust, industry-tested XML vocabulary. ACORD P&C is similar in many respects. An ACORD L&A XML document is built around a request/response model. The processing mode can be either synchronous or asynchronous (the response may include a "notify" statement as a trigger to the receiver that additional interaction is possible). The processing model is based upon a pair of messages (request and response) and uses the framework shown next:

```
<xsd:complexType name="TXLife_Type">
  <xschoice>
    <xsd:sequence>
      <xsd:element ref="UserAuthRequest"
                              minOccurs="0" />
      <xsd:element ref="TXLifeRequest"
                              minOccurs="0"
                   maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:sequence>
      <xsd:element ref="UserAuthResponse"
                             minOccurs="0" />
      <xsd:element ref="TXLifeResponse"
                              minOccurs="0"
                   maxOccurs="unbounded" />
      <xsd:element ref="TXLifeNotify"
                              minOccurs="0"
                   maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:choice>
  <xsd:attribute name="Version" type="xsd:string" />
</xsd:complexType>
```

Requests can be submitted together within a single file or envelope as a single stream, and in this case will receive a single response stream although transaction order is not maintained. Not all responses can be generated in real time; if they cannot, a notification response message is used to alert client applications to the processing of an outstanding transaction. Notification is continuously sent with other transaction response messages by a server to a client application while the response remains outstanding.

Some examples of ACORD XML for L&A processing indicators include requests for a response correlation identifier (used to determine the matching response), requests for a response indicator (used to indicate if the requestor wants notification of outstanding delayed responses), and requests for a transmission mode indicator (to indicate how a request should be treated in the context of other requests, such as a request that is an update or a replacement of another request).

Other processing indicators define the transaction to be processed or allow a requestor to dictate the nature of the results returned by the service provider, such as the data scope to return (Object, Object and related Objects, etc.), the start record for a search, or the maximum number of records to return. Still other indicators define how to specify the relationships between the message and multiple possible attachments.

Top-Down Modeling pattern

Formal modeling patterns, in contrast with ad hoc development, are used by some organizations for developing standard messages. Such formal Top-Down Modeling patterns are emerging within vertical-industry organizations. In these patterns, message specifications are developed with increased rigor, based on some form of information model. The technology, methodologies, and tools so developed may be accompanied by training classes to ensure consistency. In such environments, a large portion of the effort is spent on defining requirements, use cases and roles, and the information model. A Unified Modeling Language** (UML**) profile is often used.

This pattern provides the opportunity for standardization of elements that may be more difficult to standardize in less formal environments. A rigorous methodology can facilitate the specification of the usage of the information, and this has ramifications on the inclusion of the information elements, their cardinality, and even their semantics. As a natural consequence of this rigor, library and registry considerations arise that play a key role in the assembly of information and the definition of usage contexts.

Health Level 7 and the Top-Down Modeling pattern

One of several organizations employing such a Top-Down Modeling pattern is the advanced Health Level 7[15] (HL7) organization, a health-care information standard in which, increasingly, XML is viewed primarily as an encoding technology rather than a source information model. This pattern holds much promise for increasing the precision of standards required to promote interoperability

between businesses, reducing ambiguity and leading to reduced complexity and cost.

Version 3.0 of the HL7 standard represents an evolution in several ways. While HL7 messages have moved to XML encoding as an "implementable technology specification" (ITS), this version of HL7 introduces the strategic Reference Information Model (RIM), the Message Development Framework methodology, and the accompanying UML profile containing hierarchical message definitions (HMDs).

These features, along with HL7's Model Interchange Format (MIF), have resulted in the generation of the implementation layer (such as XML encoding) through tooling. HL7's MIF is a set of related schemas that define the set of primary artifacts that may be developed or exchanged as part of this standard. These artifacts provide a common exchange format for use between tools and repositories. Although this methodology is central to HL7, this standard continues to evolve, and its components are not fully integrated, requiring further testing.

The HL7 Clinical Document Architecture (CDA) is a document markup standard (currently in production) that specifies the structure and semantics of clinical documents for the purpose of information exchange. The CDA document source is currently encoded in XML with "derived meaning" from the HL7 RIM. It is intended that if and when alternate implementations are feasible, future technology encodings will not be limited to XML.

Information modeling through message structure definition

Organizations like HL7 employing a Top-Down Modeling pattern (using some form of information domain model) typically specify classes of information required and the properties of those classes, including attributes, relationships, and so forth, by use of a UML profile. HL7 Version 3.0 uses data type specifications, vocabulary specifications, and a RIM to derive technology-level message specifications.

The intention of HL7's MIF is to specify an XML Schema representation consistent with its UML profile, comprised of health-system-wide information structures from which ITSes are derived to define lower-level implementation technology. HL7 defines a suite of implementation tools supporting its Top-Down Modeling pattern, which includes repository interface tools, terminology table tools, modeling and message tools, schema transformation tools, and so forth.

Schema generation uses serialized MIF models. Though the MIF is still evolving, it has already had a significant impact on the direction and development of HL7 tools and plays a central role in schema generation.

WEB SERVICES BINDINGS

In the following three subsections, we describe the approach taken for Web Services bindings by the OAGi, ACORD, and HL7 organizations.

OAGIS Web Services binding

The OAGIS BOD architecture predates Web Services specifications and is independent of any communication mechanism. It can be used with simple transport protocols such as HyperText Transfer Protocol (HTTP) and SMTP (Simple Mail Transfer Protocol), but it also can be used with more complex transport protocols such as SOAP, ebMS, or any other EAI (enterprise application integration) system.

Mapping the OAGIS BOD architecture to Web Services technology occurs largely at the WSDL abstract layer.[16] WSDL's "binding" layer is accommodated by using naming conventions to define a SOAP binding specifying a WSDL document/literal encoding style, and WSDL's "service" structure is also accommodated by naming conventions. All OAGIS mappings to Web Services use a request/response exchange pattern and define mappings for an "asynchronous push" model and a "synchronous" model.[17]

The WSDL <types> structure imports all of the envelope schemas for given business content. Business content is not authored inside the WSDL <types> structure, as shown in the following example:

```
<types>
    <xs:schema elementFormDefault="qualified"
      targetNamespace=
      "http://www.openapplications.org/oagis/8.0/ws">
       <xs:include schemaLocation=
       ".../xsd/MessageResponse.xsd" />
       <xs:include schemaLocation=
       ".../.../OAGIS/BODs/AddPurchaseOrder.xsd"/>
```

```
        ...
    </xs:schema>
</types>
```

The name of every root envelope BOD element defines a single WSDL <message> structure with a single <part>, which references the top-level element, as in the following example:

```
<message name="AddPurchaseOrder">
   <part name="Message" element="oa:AddPurchaseOrder"/>
</message>
<message name="CancelPurchaseOrder">
   <part name="Message"
         element="oa:CancelPurchaseOrder"/>
</message>
<message name="ChangePurchaseOrder">
...
```

The naming convention for a portType dictates the use of the noun name, preceded by Request, Response, or Sync, and followed by PortType. Within the portType, for asynchronous push requests and synchronous request/responses, WSDL operations are defined and named by using envelope root-element names typical of the seller side of an interaction for the given business content, as in the following example:

```
<portType name="RequestPurchaseOrderPortType">
   <operation name="AddPurchaseOrder">
      <input message="oagws:AddPurchaseOrder"/>
   </operation>
   <operation name="CancelPurchaseOrder">
      <input message="oagws:CancelPurchaseOrder"/>
   </operation>
  ...
</portType>
```

Within the portType, WSDL operations are defined by using the envelope root element names typical of the buyer side of an interaction for the given business content, as in the following example:

```
<portType name="ResponsePurchaseOrderPortType">
   <operation name="ShowPurchaseOrder">
      <input message="oaws:ShowPurchaseOrder"/>
   </operation>
   <operation name="ListPurchaseOrder">
      <input message="oaws:ListPurchaseOrder"/>
   </operation>
   <operation name="ConfirmBOD">
  ...
</portType>
```

OAGIS includes process indicators within its BOD architecture, such as a confirmBOD indicator indicating the confirmation of BOD reception, as in the following example:

```
<portType name="SyncPurchaseOrderPortType">
...
   <operation name="CancelPurchaseOrder">
      <input message="oaws:CancelPurchaseOrder"/>
      <output message="oaws:ConfirmBOD"/>
   </operation>
...
   <operation name="GetPurchaseOrder">
      <input message="oaws:GetPurchaseOrder"/>
      <output message="oaws:ShowPurchaseOrder"/>
   </operation>
   <operation name="GetListPurchaseOrder">
      <input message="oaws:GetListPurchaseOrder"/>
      <output message="oaws:ListPurchaseOrder"/>
   </operation>
...
   <operation name="ProcessPurchaseOrder">
      <input message="oaws:ProcessPurchaseOrder"/>
      <output message="oaws:ConfirmBOD"/>
   </operation>
</portType>
```

ACORD Web Services binding

Because ACORD insurance specifications predate Web Services standards, much of their interaction design and supporting process indicators are implemented within the messaging content (see the section "Wrapped Content pattern"). As a result, the ACORD Web Services binding is a mapping exercise intended to address specific requirements, given the capabilities of available horizontal standards.

An example of this is the Acord Messaging Service Version 1.2 mapping based on SOAP 1.1 (document mode), WSDL 1.1, and the WS-I 1.0 basic profile (WSI-BP). When Acord Messaging Service was built, ACORD and its membership evaluated available Web Services capabilities and adoption in the context of insurance-industry requirements with the objective of preserving compatibility with existing insurance-industry messaging standards. As a result, Acord Messaging Service Version 1.2 approached areas of concern such as document and attachment management, routing, session handling, and reliability by designing support for them in the Acord Messaging Service wrapper and interaction design itself. The Acord Messaging Service Version 1.2 approach is to support the breadth of insurance-

```
<wsdl:types>
  <schema
  targetNamespace="http://www.ACORD.org/Standards/AcordMsgSvc/1.1.0"
  xmlns="http://www.w3.org/2001/XMLSchema">

<!--Inbox port messages-->
    <xs:element name="ListInRq" type="xs:anyType"/>
    <xs:element name="ListInRs" type="xs:anyType"/>
    <xs:element name="PostRq" type="xs:anyType"/>
    <xs:element name="PostRs" type="xs:anyType"/>
    <xs:element name="StatusInRq" type="xs:anyType"/>
    <xs:element name="StatusInRs" type="xs:anyType"/>

<!--Outbox port messages-->
    <xs:element name="ListOutRs" type="xs:anyType"/>
    <xs:element name="ListOutRq" type="xs:anyType"/>
     ...
<!--Call port messages-->
    <xs:element name="CallRq" type="xs:anyType"/>
    <xs:element name="CallRs" type="xs:anyType"/>
  </schema>
  <schema targetNamespace=
     "http://schemas.xmlsoap.org/soap/envelope">
     <xs:element name="Fault" type="xs:anyType"/>
  </schema>
</wsdl:types>
```

Figure 5
Generic WSDL: types element overloaded with ACORD message structure

industry standards by providing four generic modes of interaction: one-way business-message push and pull, business-message request/response, and request/response without a business message.

The binding to SOAP is thus designed to support the defined message exchange. The approach to mapping the architecture of the ACORD insurance-industry standard to Web Services technology is evident at the WSDL abstract layer. WSDL's binding layer is accommodated by using conventions to define a SOAP binding specifying a document/literal encoding style; WSDL's service structure is accommodated by a set of predefined functions; and the WSDL types structure is used to name a "generic" element that can be overloaded with any ACORD message structure as required, as shown in *Figure 5*.

Specific element names are provided as interaction proxies and are used as a means of attaching a message specification from one of the standards:

```
<wsdl:message name="PostRequest">
  <wsdl:part element="ac:PostInRq"
    name="PostRqPart"/>
</wsdl:message>
```

The required message interaction patterns are then generically invoked by using conventions for a set of predefined ports as required to satisfy an interaction with a partner. The convention for using the ports is defined in the Acord Messaging Service Version 1.2 specification. The WSDL for the ports supporting ACORD message exchange patterns is shown in *Figure 6*.

The Acord Messaging Service Version 1.2 approach uses a combination of specific message element content and protocol design to provide support for a range of requirements. Acord Messaging Service Version 1.2 and specific protocol instructions within the specification are intended to provide indicators for support of reliable and sequenced interactions, notification capabilities, and indication of "in process" status, as well as standard request/response processing.

The application element contains information indicating the ACORD standard defining the content (L&A, P&C, or reinsurance), and the version of the associated specification that defines the content.

The following code shows how Acord Messaging Service Version 1.2 introduced SOAP body `Sender/`

```
<wsdl:portType name="AcordMsgSvcInbox">

  <wsdl:operation name="Post">
    <wsdl:input message="ac:PostRequest" name="PostRq"/>
    <wsdl:output message="ac:PostResponse" name="PostRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

  <wsdl:operation name="StatusIn">
    <wsdl:input message="ac:StatusInRequest"
     name="StatusInRq"/>
    <wsdl:output message="ac:StatusInResponse"
     name="StatusInRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

  <wsdl:operation name="ListIn">
    <wsdl:input message="ac:ListInRequest" name="ListInRq"/>
    <wsdl:output message="ac:ListInResponse"
     name="ListInRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

</wsdl:portType>

<wsdl:portType name="AcordMsgSvcOutbox">

  <wsdl:operation name="Retrieve">
    <wsdl:input message="ac:RetrieveRequest"
     name="RetrieveRq"/>
    <wsdl:output message="ac:RetrieveResponse"
     name="RetrieveRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

  <wsdl:operation name="StatusOut">
  ...
  </wsdl:operation>

</wsdl:portType>

<wsdl:portType name="AcordMsgSvcCall">

  <wsdl:operation name="Call">
    <wsdl:input message="ac:CallRequest" name="CallRq"/>
    <wsdl:output message="ac:CallResponse" name="CallRs"/>
    <wsdl:fault message="soap:SOAPFault" name="SOAPFault"/>
  </wsdl:operation>

</wsdl:portType>
```

Figure 6
WSDL for ports supporting ACORD message exchange patterns

`Receiver/MsgId` elements to define the routing context for a message:

```
<ac:PostRq>
  <ac:Sender>
    <ac:PartyId>urn:duns:123456789
    </ac:PartyId>
    <ac:PartyRoleCd>11
    </ac:PartyRoleCd>
  </ac:Sender>
  <ac:Receiver>
    <ac:PartyId>urn:duns:912345678
    </ac:PartyId>
    <ac:PartyRoleCd>87
```

```
      </ac:PartyRoleCd>
   </ac:Receiver>
   <ac:MsgItem>
      <ac:MsgId>01ff00c1-e48e-4cc5-b26c-8fc210
      </ac:MsgId>
      <ac:MsgTypeCd>TXLifeRequest:204
      </ac:MsgTypeCd>
   </ac:MsgItem>
</ac:PostRq>
```

Content packaging is of particular concern. Insurance business interactions generally require documentation, which can be varied and copious. Acord Messaging Service Version 1.2 aggregated the requirements from the wrapped specifications by providing a `WorkFolder` construct that acts as a manifest for the set of attached messages, as seen in the next example:

```
<ac:PostRq>
   <ac:WorkFolder>
      <ac:MsgFile>
         <ac:FileId>cid:A01EFAE7-5490-43D0-DC
         </ac:FileId>
         <ac:FileFormatCd>text/xml
         </ac:FileFormatCd>
      </ac:MsgFile>
   </ac:WorkFolder>
</ac:PostRq>
```

Security functions in ACORD Web Services bindings use the standard SOAP recommendations. However, given the Acord Messaging Service Version 1.2 approach, ACORD created an additional set of security profiles that describe different levels of protection.[18] These profiles describe usage of Acord Messaging Service Version 1.2 extensions for the ACORD Referred Message Signature, the ACORD file digest, the ACORD file signature, and the ACORD file cipher.

Aspects of messaging and the infrastructure binding layer in HL7

HL7's messaging components exemplify the infrastructure binding layers within a Top-Down Modeling pattern. Just as high-level HMDs are exposed as XML, lower levels of messaging detail are similarly defined. At this level, the methodology includes storyboards and use cases, which define the application roles and the purpose of the information exchanged between health-care applications, and trigger events defining what prompts information exchange. Defining the information message content in the context of an ITS (e.g., the XML ITS), the required set of classes, attributes, and associations are used to develop an HMD, which provides a base template for a specific "message type"—essentially a unique set of constraints for an XML schema.

A message using the XML ITS is wrapped in SOAP and transported using secure HTTP over the Internet. The use of SOAP+WSDL is defined in a Web Services profile.[19] HL7 has announced the approval of Draft Standards for Trial Use (DSTUs) built around the HL7 RIM and has clarified its methodology from modeling to message definition and ultimately to an XML syntax representation.

Use of WSDL with the HL7 XML ITS for early industry prototypes revealed tooling issues due to the complexity of the XML ITS schemas. Typical Web Services tooling implementations using stubs and skeletons (i.e., software that is generated from service descriptions) result in the total encompassing of the content along with any ITS transmission wrappers. This is often not consistent with the domain-level application's programming model, which is not concerned with transmission wrapper semantics or processing of its constructs. This highlights the difficulties and ramifications of using differing approaches in industry-specific specifications of process indicators, such as acknowledgements, which are not specific to any industry and can be accommodated either by infrastructure technologies or a widespread higher-level standard. In the future, this may be done by a common cross-industry envelope architecture. Increased commonality in the area of content wrappers and envelopes across industries will provide off-the-shelf and open-tooling packages with increased efficiency for domain application development.

Because the Top Down Modeling pattern lacks a layered processing model and support for multiple constraint mechanisms, it is inevitable that customized code will be used when infrastructure bindings are implemented. The use of wrappers and envelopes at the technology layer does not cleanly match the classic usage of the Web Services infrastructure tooling implementation built around stub and skeleton code generation. This presents a challenging area for future work in both the information-modeling layer architecture and the infrastructure.

CONCLUSIONS AND DIRECTIONS

We have presented a set of emerging design patterns within vertical-industry standards organizations. Experience, heritage, organizational principles, established standards, support for deployed products, and changing technology all play a role in defining differences between these patterns and the organizations using them, and defining how the patterns map to the evolving Web Services infrastructure. Although these patterns have, in some cases, significant differences that must be accommodated and considered when used in a Web Services environment, their small number (four) is encouraging. It is most likely that the number of patterns that will emerge across industries will be of the same order, suggesting that Web Services provides a viable strategy for a commercial infrastructure for industry-level standards organizations. Continued examination of patterns associated with industry-level standards development provides a unique opportunity for the Web Services infrastructure, because the resulting pattern-related best practices hold the potential for providing a standardized means of optimizing the adoption of the Web Services infrastructure.

**Trademark, service mark, or registered trademark of Massachusetts Institute of Technology, Object Management Group, Inc., Open Applications Group, Inc., or OpenTravel Alliance, Inc. in the United States, other countries, or both.

CITED REFERENCES

1. World Wide Web Consortium, http://www.w3.org.
2. *Extensible Markup Language (XML) 1.0*, W3C Recommendation (February 1998), http://www.w3.org/TR/1998/REC-xml-19980210.
3. Open Applications Group, http://www.openapplications.org/index.htm.
4. Association for Cooperative Operations Research and Development, http://www.acord.org/home.aspx.
5. *ACORD Messaging Service XML Specification and SOAP Implementation Guide Version 1.2.0,* Association for Cooperative Operations Research and Development (April 2005).
6. R. Bilorusets, D. Box, L. F. Cabrera, D. Davis, D. Ferguson, C. Ferris, T. Freund, M. A. Hondo, J. Ibbotson, L. Jin, C. Kaler, D. Langworthy, A. Lewis, R. Limprecht, S. Lucco, D. Mullen, A. Nadalin, M. Nottingham, D. Orchard, J. Roots, S. Samdarshi, J. Shewchuk, and T. Storey, *Web Services Reliable Messaging Protocol* (February 2005), http://specs.xmlsoap.org/ws/2005/02/rm/ws-reliablemessaging.pdf.
7. *Business Object Document Architecture, OAGIS Release 6.2,* http://lists.ebxml.org/archives/ebxml-transport/200003/doc00006.doc.
8. *Simple Object Access Protocol Specifications*, World Wide Web Consortium http://www.w3.org/TR/soap12-part1/.
9. The MedBiquitous Consortium, http://www.medbiq.org/.
10. *MedBiquitous Web Services Design Guidelines, Version 1.0*, Medbiquitous Technical Steering Committee (April 2004), http://www.medbiq.org/technology/tech_architecture/webservicesguidelines.pdf.
11. *Web Services Interoperability Organization Basic Profile 1.1*, Web Services Interoperability Organization (August 2004), http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html.
12. Open Travel Alliance, http://opentravel.org/.
13. *OpenTravel Alliance Release OTA2005A Common Types Schema—OTA_CommonTypes.xsd*, OpenTravel Alliance, Inc. (2005), http://www.opentravel.org/2005A/OTA_CommonTypes.xsd.
14. *OASIS ebXML Message Service Specification Version 2.0*, Organization for the Advancement of Structured Information Standards (April 2002), http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf.
15. Health Level Seven, http://www.hl7.org/.
16. *Web Services Description Language 1.1*, World Wide Web Consortium (March 2001) http://www.w3.org/TR/wsdl.
17. *OAGIS Web Services Work Group*, Open Applications Group (October 28, 2003), http://www.openapplications.org/wg/WebServices.htm.
18. *Security Profiles for the ACORD Messaging Service Version 1.0.0,* Association for Cooperative Operations Research and Development (April 2005).
19. R. Ruggeri, M. de Graauw, L. F. Cabrera, M. Regio, G. Grieve, A. Julian, J. Larson, D. Pratt, and R. Spronk, *HL7 Version 3 Standard: Transport Specification—Web Services Profile, Release 2*, http://www.hl7.org/v3ballot/html/infrastructure/transport/transport-wsprofiles.htm.

Accepted for publication December 16, 2005.
Published online May 18, 2006.

Scott R. Hinkelman
IBM Software Group, 11501 Burnet Road, Austin, Texas 78758 (srh@us.ibm.com). Mr. Hinkelman is a senior software engineer whose work is focused on industry-level standards organizations and alignment with service-oriented architectures (SOAs). He serves on IBM's Emerging Technology team, helping set strategy in engagements with industry organizations and has been working on service-oriented software for over five years. While working with the OpenTravel Alliance, Mr. Hinkelman was elected to its interoperability committee and provided the technical foundation for the initial definition of XML B2B message architecture for the travel industry. He has served as the Chief eBusiness Architect for IBM's Travel Industry Solution unit. He is currently the MedBiquitous.org Web Services architect, serving on its technical steering committee, and the leader for the quality of service area in RosettaNet's WS-I Web Services profile work. He has been instrumental in many industry-wide and international standards organizations and initiatives. An accomplished Java and XML expert, he represented IBM for the JAX-RPC 1.0 specification, which defined the client programming model for Java Web services. He has published numerous articles, chaired standards conferences, and holds patents in distributed computing. Mr. Hinkelman's interests are focused on consistency in SOA design across industry standards.

Donald Buddenbaum
IBM Software Group, 4205 South Miami Blvd, Durham, North Carolina 27703 (buddenba@us.ibm.com). Mr. Buddenbaum

is a software engineer concentrating on emerging standards and vertical-industry standards organizations. He has helped leverage IBM middleware as the basis for finanical service solutions, serving for a time as the Chief Architect for IBM's Software Group insurance solutions. Before joining IBM, he spent time designing and implementing solutions in the life insurance industry at various independent software vendors and insurance companies. His current work targets the adoption of horizontal standards within vertical-industry standards organizations, such as ACORD.

Liang-Jie Zhang
IBM Research Division, 19 Skyline Drive, Hawthorne, New York 10532 (zhanglj@us.ibm.com). Dr. Zhang is a research staff member and the chair of the Services Computing Professional Interest Community at the Watson Research Center. He has been leading service-oriented architecture (SOA) services research since 2001. He was the Chief Architect of industrial standards at IBM. He has filed more than 30 patent applications in the areas of e-business, Web Services, rich media, data management, and information appliances and has published more than 80 technical papers in journals, book chapters, and conference proceedings. Dr. Zhang chairs the IEEE Computer Society Technical Committee on Services Computing and serves as editor-in-chief of the International Journal of Web Services Research (JWSR), which has been included in the Engineering Index Compendex database since 2005. He was the general co-chair of the 2005 IEEE International Conference on Web Services (ICWS 2005) and the 2005 IEEE International Conference on Services Computing (SCC 2005). Dr. Zhang received a B.S. degree in Electrical Engineering from Xidian University in 1990, an M.S. degree in electrical engineering from Xi'an Jiaotong University in 1992, and a Ph.D. degree in computer engineering from Tsinghua University in 1996. ■

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Decision Support Systems 40 (2005) 107–127

Decision Support Systems

www.elsevier.com/locate/dsw

On-demand business collaboration enablement with web services

John Y. Sayah[a], Liang-Jie Zhang[b,*]

[a] IBM Software Group, 17 Skyline Drive, Hawthorne, NY 10532, USA
[b] IBM T.J. Watson Research Center, 1101 Kitchawan Road, Route 134, Yorktown Heights, NY 10598, USA

Available online 19 June 2004

Abstract

Businesses are increasingly outsourcing key operations and interacting with ever extending nets of partners. Running extended business-to-business (B2B) operations creates the need for more advanced human interaction while advancing the automation base of B2B functions. In this paper, we introduce a model for on-demand business process-based collaboration, namely, Extended Business Collaboration (eBC), and its major elements of modeling, and a configurable business protocol-enabling framework. We discuss some of the major research issues associated with facilitating extended business collaboration, and present our proposed Annotated Business HyperChain technology leveraging Web services and semantic annotation model. A research prototype is described, followed by some observations and discussion of open research issues requiring further exploration.

Keywords: Extended business collaboration; Web services; Semantic annotation model; Business process; Annotated Business HyperChain

1. Introduction

Enterprises are not standalone anymore. They need to work with their value net of partners and customers. Whether we are considering, logistics, financial services, suppliers, handling customer orders, or marketing programs, large or small enterprises operate and interact in variety of forms within a complex global web of collaborating entities. A 21st century enterprise, not requiring all of its needs to be fulfilled by internal groups, leverages some well-proven services and available products in its daily operation processes and in product designs. Taking a new product design as an example, a consumer electronics company has to work with component suppliers, electronic manufacturing services (EMS), or contract manufactures to design a product collaboratively. Some component design work may be outsourced to design partners who are specializing in special components such as Application Specific Integrated Circuit (ASIC) chips, batteries, or motherboards. Outsourcing non-core-competency services has become a popular trend fueled by business transformation drivers. Information technology services and high-tech product developments are at the forefront of this trend that is becoming the dominant business model in the era of globalization. It is the outsourcing model that enables disaggregated businesses to form a value chain for creating more innovative and higher quality products or services than that they would have accomplished by their own.

In a typical business value chain, the trading partners or design partners could be dynamically added or removed in the lifecycle of a business solution, operation or during product development

* Corresponding author.
E-mail addresses: john_sayah@us.ibm.com (J.Y. Sayah), zhanglj@us.ibm.com (L.-J. Zhang).

0167-9236/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2004.04.011

or servicing thereafter. All the resources including business entities, services provided by business entities, documents and messages would be activated and accessed in what could be dubbed as an ''on-demand model''. In this on-demand model, a service is ''brought-on-line'' if and when is needed and solely for the duration of its need. In this model, the goal is to maximize efficiency and productivity. A possible transformation can now take place, from entities interacting in a disaggregated setting to one of a virtual on-demand enterprise operating in extended business collaboration mode. In this mode, requesting components for a product from a supplier would seamlessly locate the best suppliers but would concurrently enable and manage the related financial, logistics, or legal services. In this mode, we would also move from the paradigm of granular commercial transactions, such as issuing a purchase order (PO), to a paradigm of integrated business processes. These extended business and collaboration processes that engage and involve multiple business entities concurrently now have an extended lifecycle that is dynamic and cannot be confined and solely defined by a simplistic model of transactional handshakes or totally depend on the human intervention to manage it.

Fully automated business collaboration remains a goal on the horizon. However, experience leads us to believe that people-assisted enabling technologies can pave a way to this realization. Leaving aside practical business and geo-political considerations, preventing some companies from fully embracing extended business collaboration, we will attempt in this paper to address some of the key technical obstacles. We then present a methodology, supporting framework and techniques that would enable the practical realization of this on-demand collaborative business vision.

From a technical viewpoint, the real obstacles go beyond business process representation and data transformation techniques. The real problem arises from the fact that we are dealing with interactions between two or more business entities and their loosely coupled business processes. These business processes could be private business processes in some enterprises or public processes crossing the boundary of multiple enterprises. In this environment, the workflow is non-deterministic and you have projects and operations running across multiple companies with a mix of automation and human-driven actions. Timetables, project schedules, and response times are equally fluid. Leveraging emerging and evolving standards is a key starting point to help address the aforementioned challenges and problems.

Web Services [18] are network-enabled reusable components that conform to an interface with standard description format and access protocols. The basic enabling infrastructure of Web Services consists of UDDI registries, Simple Object Access Protocols (SOAP), Web Services Definition Language (WSDL), Business Process Execution Language (BPEL4WS), Web Services Inspection Language (WSIL), and so forth. Web Services provide the means to enable the integration in a standard way.

In this paper, we outline a new Extended Business Collaboration (eBC) model first. We then present, in Section 3, an enablement framework based on Web Services, followed by an illustration of a design collaboration scenario in a distributed industrial environment. Some related works are given in Section 4. The paper concludes with some observation and discusses further research topics.

2. Extended business collaboration (eBC) model

Several approaches have been proposed to represent business behaviors and a variety of ''layer'' models circulate in the business modeling domains. All models, independent of the adopted layering approach, do basically include a higher business layer and a lower Information Technology (IT) infrastructure layer. Typical business models are: Business-to-Customer (B2C), Application Service Provider (ASP), Application to Application (A2A) also called Enterprise Application Integration (EAI), and Business to Business (B2B). All of these ''classic'' models strictly differentiate between intra-enterprise interactions and inter-enterprise interactions. In support of these enterprise-based interaction models, different interaction techniques (business portals, e-mail, fax, etc.) and general and vertical industry standards (EDI, ebXML, RosettaNet [14], etc.) have emerged over the latter part of the last century providing various levels of business interaction and connectivity deployment.

As outsourcing and on-demand operation models are becoming more and more popular, the boundary between enterprises is gradually bypassed or elimi-

nated. Enterprises are collaborating by different degrees and levels. The environment—the extended enterprise, the information sharing, the functions and the processes that enable two or more parties, within and cross-business entities, to interact in the context of business activities and events to deliver business results, are all being transformed. Hence, the representation needs evolve to capture and cover this emerging business paradigm.

We believe that an Extended Business Collaboration that dissolves or rather bypasses this enterprise boundary-based distinction is a factual representation of what is evolving in the real business world. This model aims at reducing the artificial elements at the boundary inherent in the enterprise-based interaction models. The diffusion of the boundary implies that new techniques are required in order to evolve converge the various converge interaction methods. Introducing business semantic computing techniques, creating pluggable business collaboration protocols, proposing dynamic activity chain representation, and leveraging distributed project and business process management and monitoring capabilities may pave a way to on-demand business collaboration, which has a more structured but flexible collaboration adaptability.

In the case of Product Life Cycle Management (PLM), the parties interact in a dynamically established virtual team and enterprise setting, during the concept, design, build, or servicing of a product to create a more innovative, profitable, higher quality product brought to market sooner than the parties would have accomplished on their own. By bringing PLM into the ideal eBC setting, we move from ad hoc and transactional interactions to constructing, activating, tracking and, monitoring collaborative development and design processes of a product involving multiple companies or organizations inside one company. An example eBC deployment scenario is illustrated in Fig. 1.

A product company may engage an ASIC supplier, an EMS, or some other service providers at various stages of product design and development [4]. These "partners" may in-turn engage other partners down their value chain. The timing and the infrastructure supporting these interactions vary in terms of frequency, automation, and the individuals who participate in the process. Some partners are connected through B2B

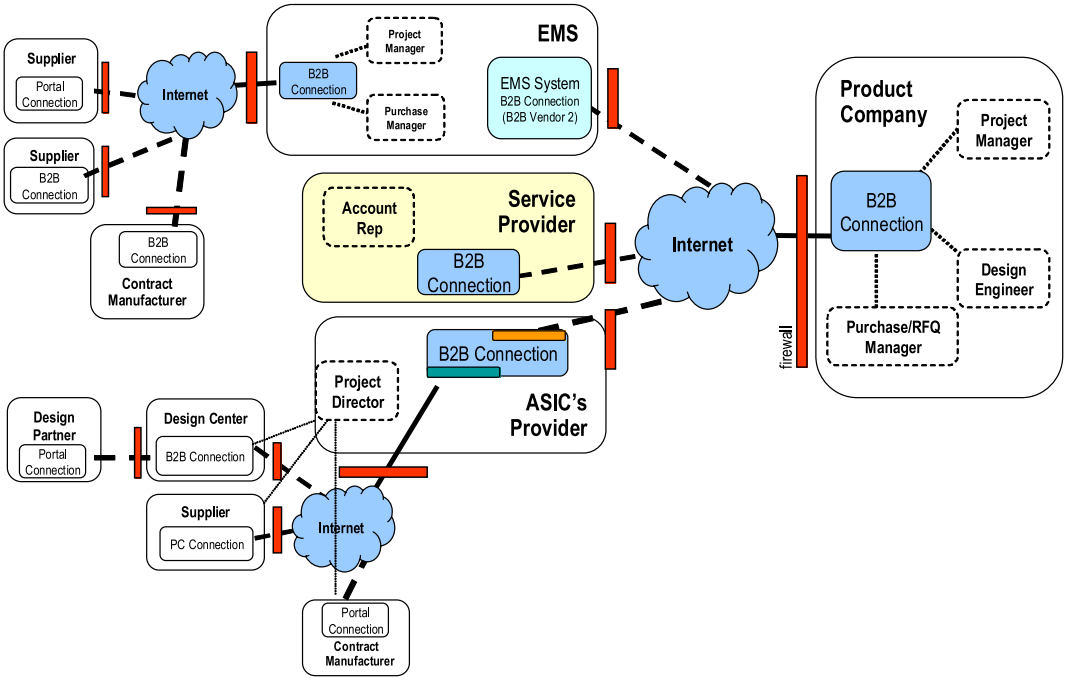


Fig. 1. Extended business collaboration deployment example for PLM.

software, to support transactional exchanges, while others may connect occasionally through a portal interface. The patterns of active connectivity and their durations and where information is in this design and supply *hyper-chain* are neither fixed nor static. This is a not a traditionally well-defined supply chain where all participants could be known in advance. Each partner knows and interacts "on-demand" with its immediate partners and information gets propagated up and down this hyper-chain. In this setting, an eBC solution should support the observation and control of related supply chain management and collaboration activities during the design and development of a product. Meanwhile, the desired eBC solution should manage the dynamics characterizing this environment including the formation and dissolution of collaborating teams and the related effects. From the solution management point of view, the eBC solution needs to monitor the design process status, design data status at any granularities, across all design and development partners, and across the individual participants. In case of business exceptions or when quick decisions are required, the desired eBC solution should provide the mechanisms to enable efficient escalation activation and timely problem resolution of issues that impact the design and development processes and schedules of a product and related business activities.

Although Web services infrastructure provides a good foundation to build such a flexible eBC infrastructure, there remains a variety of challenges to achieve these goals. One of the key challenges involves the type and nature of information exchanges. As Fig. 1 illustrates, partners in the design and supply hyper-chain do not have homogenous environments whether for internal or external operation and connectivity. Furthermore, it maybe impossible in this setting to determine a priori all levels and type of exchanges. Hence, it is imperative to provide sufficient *annotations* during an interaction process to permit the parties at the interacting edges to exchange and interpret what is required and where to access information or data. Areas requiring special attentions that drove our research activities included:

- defining a flexible annotation information representation format,
- delivering the information in a standardized or standardize-able fashion,
- interpreting, processing, and directing information exchanges based on needs,
- controlling information exchanges and flow dynamically,
- monitoring the status of process flows and documents exchanges,
- insuring and managing the security of all communication channels and information access.

Therefore, in the next section, we presents an XML, Web Services and other standards based Annotated Business HyperChain (ABH) technology for creating and managing eBC infrastructure and solutions.

3. Annotated Business HyperChain technology for eBC enablement

In this paper, we propose a new technology—Annotated Business HyperChain—that addresses the areas of semantic representation, collaborative exchange protocols, and on-demand information exchange model to enable extended business collaboration. The Annotated Business HyperChain technology consists of three major components, namely, the *eBC Ontology*, the *Collaborative Exchange Protocol* (CxP), and the *HyperChain Manager*. Annotated Business HyperChain technology also extends the concept of hyperlinks of an object in HTML files to the solution components and the resources involved in a business collaboration chain.

The eBC ontology, defining the commonly shared knowledge regarding the business semantics of the information that get exchanged during business collaboration, provides the foundation for understanding and interpreting the information involved in the process. The eBC ontology enables a flexible and uniform annotation representation for information exchanges of various non-structured, and ad hoc data without requiring pre-defined schemas. Based on the eBC ontology, CxP defines the set of elemental and composite messages that may be exchanged between two or more parties engaged in collaborative business activities. CxP is a business goal-oriented protocol supporting a wide variety of business constructs and a versatile message composition that accommodates the

needed variations in the life cycle of a collaborative business process.

A basic operation principle relating to the proposed information exchange model in this paper is to communicate schema-less HyperChain annotation data. Recipients would then follow the HyperChains to access or fetch required and detailed information, such as design files, design specifications, Bill of Materials (BOM) files, based on the roles of the recipient or their position in the business chain. These on-demand file and data transfer modes are enabled through self-retrieving or agent-based file transfer services. In the meantime, the model supports tractable information associated with design files, design processes and BOM.

In general, an on-demand information exchange model which enables implementation of the above-described operations is designed to achieve the following goals: (i) provide a flexible and uniform annotation representation for information exchange of various non-structured data without requiring predefined schemas; (ii) automate the annotation data generation process; and (iii) capture and automate business collaboration interaction patterns for information exchange based on the annotation data. Moreover, delivery policies are provided to control how the on-demand contents are to be delivered. There are at least four types of delivery models:

Scheduled content delivery—On a predetermined, periodical schedule, the information content can be delivered to the intended recipients.

On-demand content delivery—Ad hoc, based on user's request, i.e., following HyperChain specified links that provide with the annotation data and download the data. In most cases in a design collaboration scenario, the design file is very large and a server-to-server file transfer mechanism may be needed for assisting on-demand content delivery.

Access control-based content delivery—Delivering contents depending upon the role and authorization of a recipient and user credentials. As business collaboration generally involves multiple enterprises, the regular single-sign-on security mechanism has to be enhanced to incorporate annotated access control policy in the business annotation data based on entitlements. For example, who can see a document or modify it, and when it should be sent back or forwarded to other participants.

Push-based content delivery—Sending annotation data along with attachments. In general, this model is suitable for small size file transfer.

To effectively annotate the data for business collaboration, we propose an extensible data structure based on the proposed eBC ontology for data annotation, describing design collaboration processes, design activities such as design requirements, references, specifications, and design tools. The eBC ontology is built on the Resource Description Framework (RDF) [13] that provides the flexibility and versatility to support various data format required in collaboration message flow and document exchanges. A detailed description on the eBC ontology will be given later.

HyperChain Manager is the core processing engine that is responsible for creating, sending, receiving, and processing annotation messages. Additionally, it implements the on demand information exchange model and escalation process launch. It also provides an enabling platform to dynamically configure business constructs that guide the follow-on interactions between design partners. It serves as a platform to provide an extendable data aggregation mechanism to integrate information from multiple partners' data sources for effective monitoring and visibility control. As shown in Fig. 2, a diagram illustrates a HyperChain manager, which serves as a CxP engine.

The deployment architecture comprises an extended business collaboration portal/dashboard, an extended business collaboration (eBC) manager, a HyperChain manager, a B2B Gateway such as WebSphere Business Integration Connect (WBI-C) [16] layer that servers as a gateway between the HyperChain manager and the lower-level web application server such as a WebSphere [17] layer that is a typical Java-based web application server for hosting business applications. The portal or dashboard includes applications that can access the HyperChain Manager via the application-programming interface (API) layer provided by eBC manager.

HyperChain manager comprises a collaborative directory (with manager and directory), an annotation manager, a message sender, a message receiver

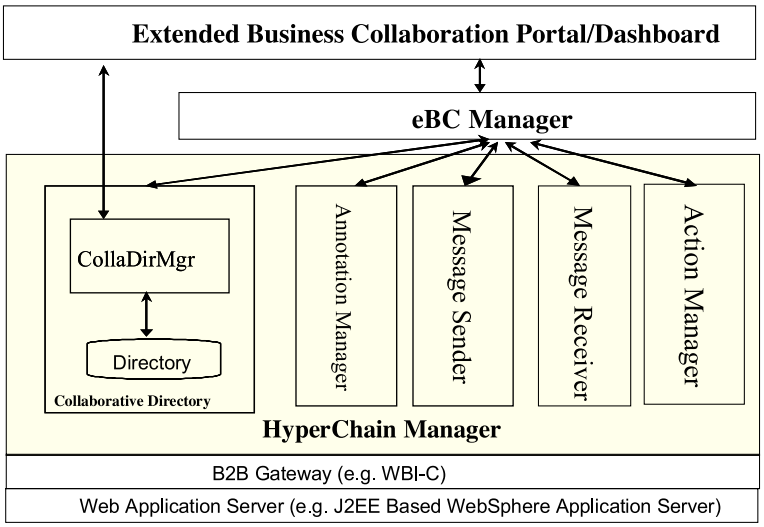


Fig. 2. HyperChain manager.

and an action manager. The collaboration directory manager component manages the resources tracked by the HyperChain Manager, such as organizations (partners), users, projects, tasks, etc., and the resources are RDF-based. The CxP messages are sent and received by the message sender and receiver modules and they are Simple Object Access Protocol (SOAP) [15] messages or other protocol messages such as Message Queuing (MQ) messages. The Message sender and Message Receiver shown in Fig. 2 have Web services interfaces. In fact, from the deployment point of view, few existing collaboration technologies or frameworks can be deployed on different application platforms (Windows, Linux) and environments (tool-specific environments). That is, platform-independent collaboration technologies have not been widely adopted in the loosely coupled business process integration domain. In this paper, we leverage the emerging Web services technology to create platform-independent interfaces to support flexible information exchange across multiple enterprises.

The annotation manager processes the meta data or annotations created for the documents and information exchanged via CxP messages. Examples of annotations are file name, file type, version, author name, etc. In addition, annotations can also be used to specify "actions" to be performed on the documents. Examples of such actions may be "review" document, perform "RFTP" (reliable file transfer) and send actions to legacy applications like Enterprise Resource Planning (ERP) and Product Data Management (PDM), etc.

The annotations in the received messages are forwarded to the action manager, which is an integration layer to back-end legacy applications as well as components like RFTP. The action manager invokes the proper actions on the documents.

As shown in Fig. 3, a "collaborator (1, 2,... *M*, *N*,...)" is a business entity that participates in a business collaboration process with one or more external business entities. "App" (l, 2, 3,...) denote the backend applications that the HyperChain manager integrates with the existing business process through an action manager component. The "collaborative directory" stores the resources of the business collaboration, such as projects, tasks, users, organizations, documents, as well as annotations/meta data that are managed by the HyperChain manager. The "HyperChain dashboard" in Fig. 3, is a graphical user interface (GUI) providing management and monitoring functions through which people interact with the collaboration resources stored in the collaborative directory.

The sections that follow describe in detail main components of Annotated Business HyperChain architecture and the underlying technology components.

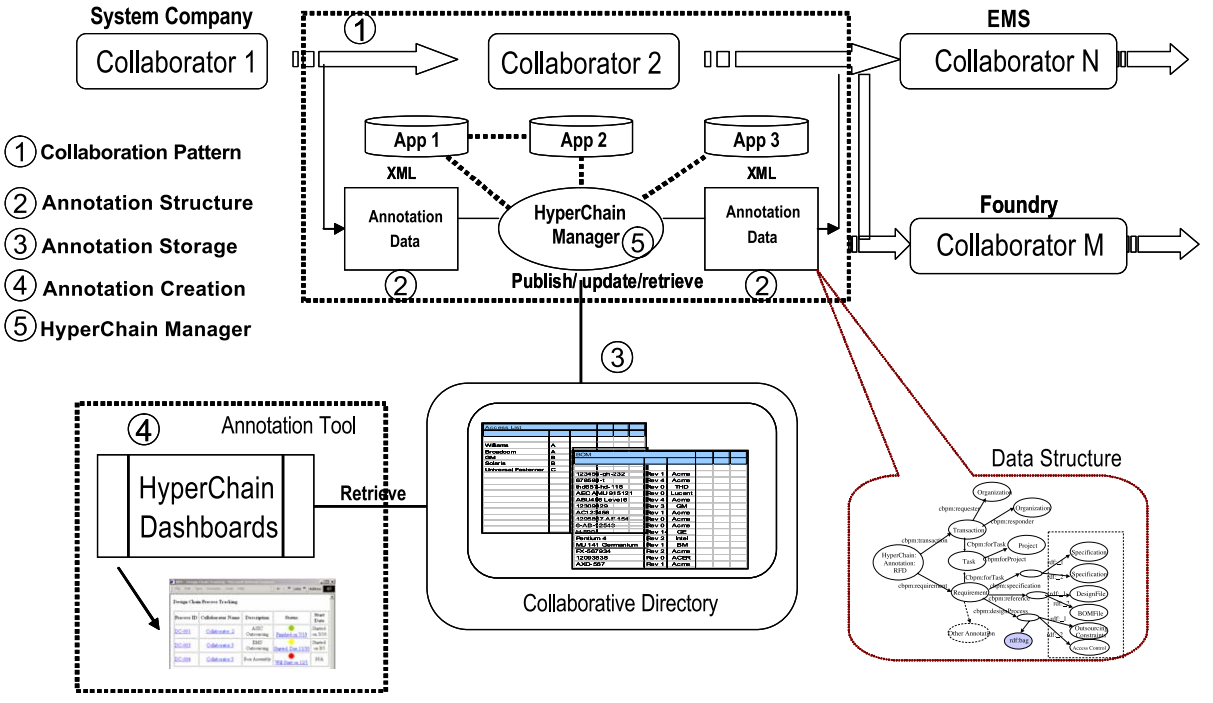


Fig. 3. HyperChain information propagation architecture.

3.1. Semantic annotation model for eBC

An ontology or commonly shared knowledge defines the business semantics to annotate information to be exchanged. An ontology is required for providing a foundation for business collaboration. Without such shared common knowledge, participants will not be able to decipher the exchanged information. Existing collaboration solutions are usually based on fixed knowledge pre-configured at each collaborator side reducing flexibility and functional scalability of collaborative activities.

The semantic annotation model for eBC, referred as eBC Ontology, which provides a flexible and uniform annotation representation for information exchanges of various non-structured, ad hoc data without requiring pre-defined schemas. We use Resource Definition Framework (RDF) to capture all three types of semantic representations, namely, Organizational Behaviors, Data Entities for Business Collaboration, and Activity Ontology for Extensible Application Integration. eBC Ontology is an instance of RDF schema [3] for extended business collaboration. It consists of all three semantic representations listed above to address the organization behaviors, external activities as well as individual resources in the context of eBC. An example Request for Design annotation RDF graph is shown in Fig. 4.

As shown in Fig. 4, all collaborators use the basic ontology to exchange business information. The business collaboration ontology uses a RDF model for specification purposes. Annotation is one part of the ontology. For example, we define RDF resources such as "Site," "Organization," "Project," "Task," "Requirement," "Transaction," "Documents," "Annotation," etc. in an RDF schema. The RDF schema serves as the basic ontology definition that all collaborators need to understand.

An RDF-based business collaboration ontology treats all newly added entities as resources. Thus, the same mechanism used to handle existing resources may be used to handle newly added resources. The extensive and flexible features of the business collaboration ontology allow the definition of any annotations without being restrained by the schema of the annotation data. Thus, as mentioned above, Hyper-

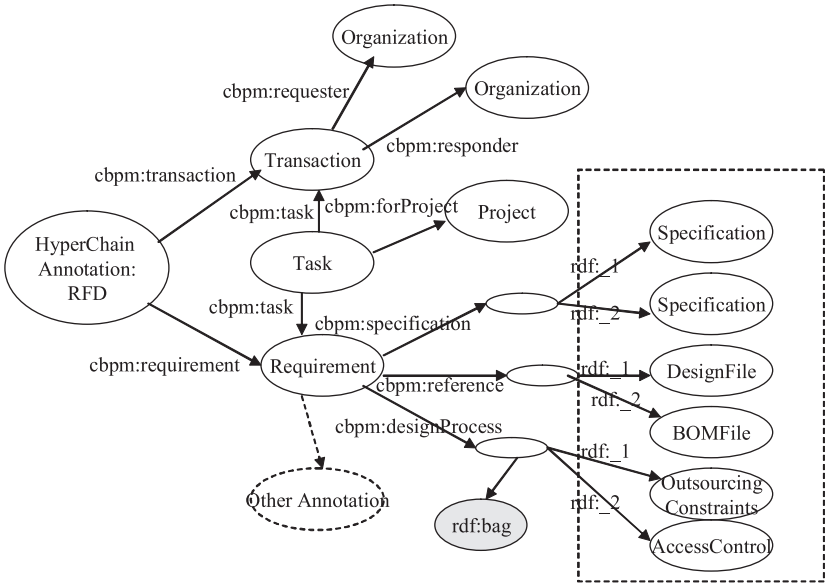


Fig. 4. RDF graph for HyperChain annotation data.

Chain annotation data that conforms to the business collaboration ontology is effectively schema-less.

In our approach, the ontology definitions are the basis that the collaborators need to understand and agree upon before exchanging information. Some example resources of the ontology currently provided by eBC are: Site, Product, Part, DesignComponent, Service, Party, Organization, Individual, Project, Task, OutsourcingTask, HomeTask, Message, Document, DesignFile, Requirement, Specification, Reference, Status, etc.

In addition, each collaborator can define their own ontology (Extended Ontology) and add additional annotations into the basic ontology for their own special needs. For instances, we can use fileName, fileSize, and format to annotate a specific design file. Again, these extended ontology or annotation definitions also need to be propagated to business partners prior to business exchanges take place. eBC Ontology is the collection of the basic and extended ontology, which is used by HyperChain Manager, which will be described in detail later, to create resources and model.

As shown in Fig. 5, a diagram illustrates a hierarchical annotation structure of entity classes defined for a HyperChain manager.

In Fig. 5, root class is the Site, which can be associated with zero or more Organization classes, representing business entities. Each Organization class can be associated with zero or more Project classes, each of which in turn can be associated with zero or more Task classes, as well as zero or more PeopleCollab utilities. Each Task class can be associated with zero or more Transaction classes, which in turn can be associated with zero or more (CxP) Message classes. In addition, each Task class can be associated with zero or more Requirement classes, representing requirements to be sent to the partners. Each Requirement class can be associated with zero or more Annotation (which may include meta data to describe the requirement), Specification, and Reference classes.

Annotation Property is the Java class that the actual annotations are created from, e.g., filename, authorname, price, etc. Hence, the relationship indicates "use". "PeopleCollab" refers to the agent or broker that conducts a human collaboration process, which is part of the extended business collaboration process. The example human collaboration process may be launching a chat program, creating a discussion thread in a discussion forum, and so forth. "0...*" in Fig. 5 means that the association relationship is 0 or more,

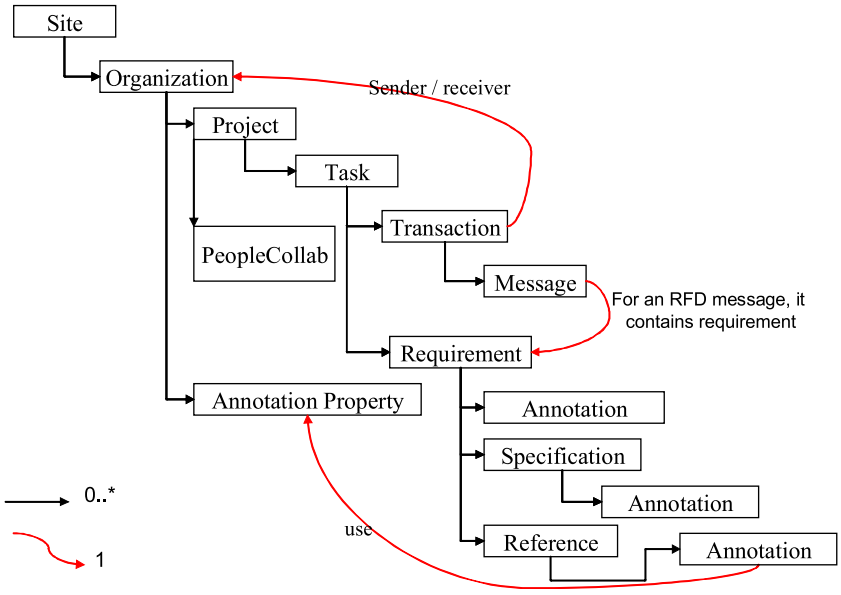


Fig. 5. Class diagram of the hierarchical annotation structure of eBC.

i.e., source class can be associated with zero or more instances of the target type where the straight arrow (→) is pointing to. "1" refers to the association relationship being one to one.

An example annotation data instance is shown in Table 1. It shows a sample business process annotation data instance for an RFD message of the entire design chain. The "transaction" resource defines the exchange context of the messages, RFDTransaction, the requester, PDT, and the responder, MyComputerCorp. The "Task", T61MotherBoardDesign, and "Project", T61BoardDesign, resources define which project and task the messages are bound to. Several containers are defined to group the following annotations or metadata: Design specification annotation, T61Specification.pdf, Design file annotation, T21MotherBoard DesignFile.cat, BOM file annotation, T21BOMFile.bom, Design process annotation, outsourcingConstraint, and other related annotations, such as access control. Note that "cbpm" stands for "Collaborative Business Process Model" we defined in this paper. "rdf" stands for "Resource Definition Framework".

The design activity annotations mainly focus on constraints in the design collaboration process. There are several types of activity annotations, e.g., design requirements, design configurations/specifications, the design files, BOMs, design processes, etc., and each one is for a different purpose. Thus, each contains different annotation data with a different format. However, they should all follow the same design rule. The following example in Table 2 shows a design activity annotation where several constraints are specified as well as the access control using OASIS eXtensible Access Control Markup Language (XACML) [11] to express the constraints.

3.1.1. Annotation storage: collaborative directories

We continue to use design collaboration to illustrate eBC concepts. The design collaboration ontology is defined in RDF schema format, and is stored in RDF format. Annotation is one part of the ontology. There are diverse requirements for annotation in design collaboration, and new requirements emerge endlessly. In addition to the pre-defined annotations for electronic business collaboration, users can define custom annotations. Table 3 is a sample of storage of an annotation definition.

These annotation definitions are applied to various elements during the information exchanges in design collaboration processes. An example RDF representation request for design (RFD) message during the RFD primitive for a design project will be illustrated in details later.

Table 1
RFD message

```
<rdf:RDF
<cbpm:RFD rdf:ID="007">
        <cbpm:transaction>
        <cbpm:Transaction  rdf:about=
"http://www.pdt.com/dc/directory/transaction#007001"_>
        <rdf:type>&cbpm;RFDTransaction</rdf:type>
        <cbpm:requester
rdf:resource="http://www.pdt.com/dc/directory/Organization#PDT"/>
        <cbpm:responder
rdf:resource="http://www.pdt.com/dc/directory/Organization#MyComputerCorp"/>
        <!--anything about the Transaction-->
        </cbpm:Transaction>
        </cbpm:transaction>
        <cbpm:task>s
        <cbpm:Task
rdf:about="http://www.pdt.com/dc/directory/task#T61MotherBoardDesign">
        <!--anything about the task-->
        <cbpm:forProject>
        <cbpm:Project
rdf:about="http://www.pdt.com/dc/directory/project#T61BoardDesign">
        <!--anything about the project-->
        </cbpm:Project>
        </cbpm:forProject>
        </cbpm:Task>
        </cbpm:task>
        <cbpm:requirement>
        <cbpm:Requirement
rdf:about="http://www.pdt.com/dc/directory/requirement#00d034334">
        <!--anything about the requirement-->
        <cbpm:specification>
        <rdf:Bag>
        <rdf:li>
        <cbpm:Specification  rdf:about="http://www.pdt.com/pdf/T61Specification.pdf"/>
        </rdf:li>
        </rdf:Bag>
        </cbpm:specification> <cbpm:reference>
        <rdf:Bag>
        <rdf:li>
        <cbpm:DesignFile
rdf:about="http://www.pdt.com/pdf/T21MotherBoardDesignFile.cat"/>
        </rdf:li>
        <rdf:li>
        <cbpm:BOMFile  rdf:about="http://www.pdt.com/pdf/T21BOMFile.bom"/>
        </rdf:li>
        </rdf:Bag>
        </cbpm:reference>
        <cbpm:designProcess>
        <rdf:Bag>
        <rdf:li>
        <cbpm:OutsourcingConstaints
```

Table 1 (*continued*)

```
rdf:about="http://www.pdt.com/outsourcingConstraint"/>
        </rdf:li>
        <rdf:li>
        <cbpm:AccessControl rdf:about="http://www.pdt.com/accessControl"/>
        </rdf:li>
        </rdf:Bag>
        </cbpm:designProcess>
        </cbpm:Requirement>
        </cbpm:requirement>
</rdf:RDF>
```

The annotation can be stored in a collaborative directory, which can be deployed on each enterprise site. Collaborative Directory stores all the HyperChain annotation data and partner's profiles as well as the links to different other data sources accessed by participants in a design chain. The collaborative directory consists of Web services utilities and a relational database or plain XML file for storing the collaborative data. At the same time, the collaborative directory provides Web services utilities that used to populate services for updating/publishing data; monitor the status of the services at different levels and for eBC dashboard. It acts as a File Transfer Agent to invoke the file transfer service on B2B collaboration environment; as well as to connect with HyperChain Manager.

Since the data with embedded status information (e.g., about a project, tasks, exchanged documents, etc.) are stored in multiple collaborative directories, the information from these distributed collaborative directories can be aggregated based on an access control policy carried in the annotation data. Another deployment of the collaborative directory is to act as a hub where the hub manages collaborative resources of multiple organizations that use the hub as a central repository in support of collaboration activities.

3.1.2. Annotation creation and portal integration

Annotation creation is a major function of the eBC enabling platform. It may be performed with the assistance of annotation tools. As mentioned above, all the annotation data of various resources used in business collaboration are stored in annotation storage, collaborative directories, such as plain text files or relational databases. The annotation creation process may operate on the storage to create annotations. In general, creation of annotation includes the following steps:

(1) Collect information by use of extended business collaboration portal or other GUI interfaces. The information includes the description of various resources such as partners, projects, tasks, specification annotations, reference design file annotations, and other related annotations.
(2) Store all the collected information into the annotation storage.
(3) Extract required data from the storage to organize the annotation message to be exchanged.

Let us take the RFD message creation as an example to illustrate the process. First, the user creates a new task as an outsourcing task or internal

Table 2
Design activity annotation

```
<cbpm:design-activity-annotation>
<cbpm:desc>
<cbpm:checkpointConstraints>. . . </cbpm:checkpoint constraints>
<cbpm:outsourcingConstraints>... (XACML) </cbpm:outsourcingConstraints>
<cbpm:acl>      ... (XACML)     </cbpm:acl>
<cbpm:documentFormatConstraints>http://temporg.com/tempuri/documentFormatContraints.htm
</cbpm:documentFormatConstraints>
</cbpm:desc>
</cbpm: design-activity-annotation>
```

Table 3
Sample RDF schema for HyperChain annotation

```
<rdf:RDF
  xmlns:daml='http://www.daml.org/2001/03/daml+oil#'
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'>
  <daml:DatatypeProperty rdf:about='http://www.ibm.com/ibm/pdt#fileName'
    rdfs:label='fileName'>
    <rdfs:comment>The name of a file, huh?</rdfs:comment>
    <rdfs:domain rdf:resource='http://www.w3.org/2000/01/rdf-schema#Resource'/>
    <rdfs:range rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
    <rdfs:isDefinedBy rdf:resource='urn:Organization:PDT@PDT'/>
  </daml:DatatypeProperty>
  <daml:DatatypeProperty rdf:about='http://www.ibm.com/ibm/pdt#fileSize'
    rdfs:label='fileSize'>
    <rdfs:comment>The size of a file</rdfs:comment>
    <rdfs:domain rdf:resource='http://www.ibm.com/cbpm#Document'/>
    <rdfs:range rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
    <rdfs:isDefinedBy rdf:resource='urn:Organization:PDT@PDT'/>
  </daml:DatatypeProperty>
  <daml:DatatypeProperty rdf:about='http://www.ibm.com/ibm/pdt#format'
    rdfs:label='format'>
    <rdfs:comment>The format of a file, huh?</rdfs:comment>
    <rdfs:domain rdf:resource='http://www.ibm.com/cbpm#Document'/>
    <rdfs:range rdf:resource='http://www.w3.org/2000/10/XMLSchema#string'/>
    <rdfs:isDefinedBy rdf:resource='urn:Organization:PDT@PDT'/>
  </daml:DatatypeProperty>
</rdf:RDF>
```

task. Then, the user may specify design requirements for the design task. The requirements may include specifications, reference design files, design process constraints, etc. Fig. 6 illustrates the creation of a Request for Design (RFD) message, including design requirement annotations, specification annotations, and annotations about the related reference documents.

After the information is collected and stored in the annotation storage, the annotation creation process starts. The annotation creation module extracts required data from the storage and forms the RFD annotation message based on the eBC ontology. The generated RFD message will be sent to design partners. After receiving the RFD message, partners can view the annotation and merge it with their own annotation storage. If a partner desires to learn more about one of the annotated resources, the partner can get the annotation link (such as design file annotation link) and request more information. The sender will generate an annotation for the design file and send it back to the partner. The partner can determine whether or not to retrieve the actual design file based on the annotation. Thus, on-demand information exchange is performed.

3.2. Collaborative exchange protocols (CxP)

Based on the eBC ontology, CxP uses RDF to annotate business collaboration processes by defining industry specific ontology, allowing peer-to-peer interaction between collaborative processes. CxP is a typical collaboration pattern in Fig. 3. CxP comprises of the messages to be exchanged between two parties or among multiple parties, some predefined message exchange sequences, and a set of business goal-oriented protocols composed by some predefined message exchange sequences. CxP builds on top of a set of standard protocols and adds the features needed for extended business collaboration processes. CxP are used to transmit the semantic representation and control the information exchange flow as well as monitor the on-going activities in a dynamic fashion.

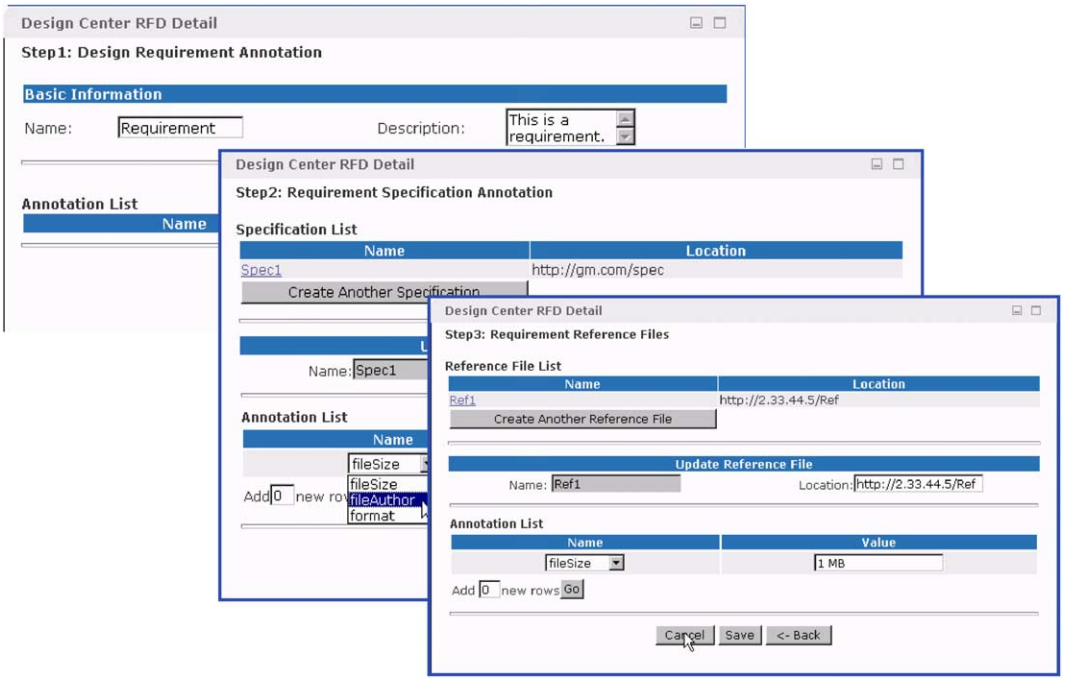


Fig. 6. Flexible CxP message creation using web portal.

As shown in Fig. 7, the collaborative exchange protocol stack of eBC supports elements of various granularities, which are configurable building blocks for creating adaptive solutions to achieve a business goal. As shown, in the protocol architecture, the following elements are defined: business scenario, business constructs, collaboration primitives, messaging layer and transport layer. The corresponding descriptions on each layer are multiple business constructs, multiple primitives (e.g., request for design (RFD) primitive and design submission (DS) primitive), multiple CxP messages (e.g., RFD primitive, DS primitive), CxP message with HyperChain annotation, and standard transport protocols. In the

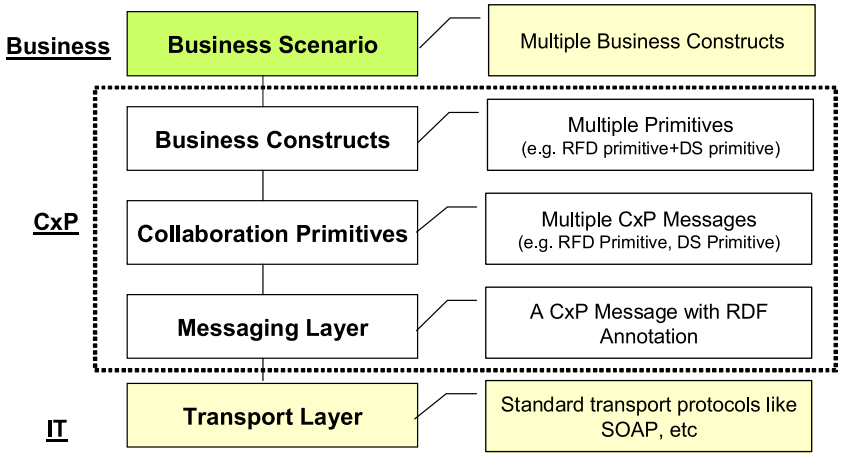


Fig. 7. Collaborative exchange protocol (CxP) stack.

messaging layer, RDF is used to represent business collaboration annotation. On top of the messaging layer, a set of primitives is defined as collaboration primitives for communication and collaboration between the parties.

A business construct is a basic unit of message exchange sequences which serve a single business goal. For example, an RFD business construct is used when a request for design is initialized, e.g., a design center, Product Design Team (PDT), shown in Fig. 8, can send RFDs to its design partners to do motherboard designs or to do mechanical and electrical designs. Following that, an Accept or Reject primitive may be received from the design partners. A business scenario serves a more complex business goal-like design-outsourcing scenario. Each business scenario may comprise several business constructs depending on the corresponding business context.

Collaboration primitive, business construct and business scenario concepts are described in detail below.

In CxP, an atomic message is defined as a rudimentary exchange of information between collaboration partners, e.g., an RFD message. A set of choreographed messages forms a primitive. For example, RFD primitive may comprise two messages, e.g., RFDMessage and AckMessage. Furthermore, one or more primitives form a business construct. For example, RFD business construct may comprise two primitives, e.g., RFD primitive and Acceptance/Rejection primitive. Scenarios are sequences of business constructs that represent a complex interaction among business partners, such as design initialization, engineering change management, and opportunity launch. In addition, CxP primitives and business constructs are targeted for specific collaboration goals and, even though configurable, they are relatively fixed. While business scenarios can be composed in several ways and thus are quite flexible.

As we introduced earlier, a design collaboration primitive is a group of message exchanges for a specific and micro-design collaboration goal. Several core design collaboration primitives are defined for CxP: Request For Design (RFD), Accept or Reject a request (Accept/Reject), Design Submission (DS), Request For Information (RFI), Information Submission (IS), Request For Update (RFU), Update Submission (US), and so forth.

Let us take RFD as an example; each collaborator uses the RFD primitive to request a partner to

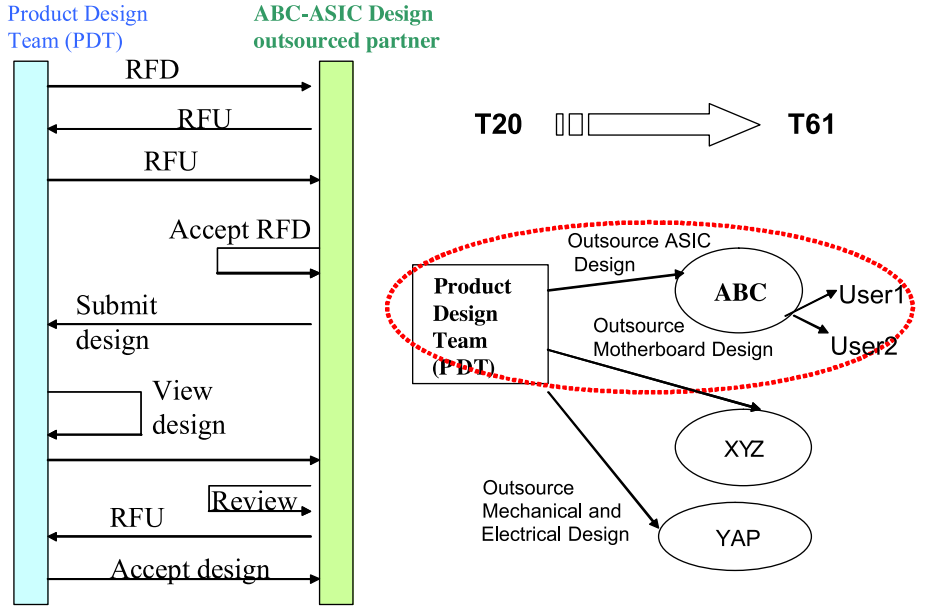


Fig. 8. Example ThinkPad design process templates.

perform a design task. An RFD primitive comprises three messages: RFD, RFD_Receipt_Ack, and RFD_Acceptance_Ack messages. This is illustrated in Fig. 9.

RFD Message: sent by the originator, e.g., a design center, to a recipient, e.g., design partner. Contains a requirement comprising specifications, references, and annotations.
RFD_Receipt_Ack Message: sent by the recipient; a response to RFD message, indicating the RFD message has been received by the recipient.
RFD_Acceptance_Ack Message: sent by the recipient, containing a flag indicating whether the recipient accepted or rejected the RFD.

Table 4 is an example of RFD Message.

Each design partner may accept or reject the request after the partner received either an RFD or RFU. One example of an Accept primitive to an RFD is illustrated in Table 5.

3.2.1. Business construct

A business construct comprises a group of collaboration primitives, which can be selectively configured for a business construct. Once configured, a business construct is organized in a relatively fixed fashion to achieve a single design collaboration goal.

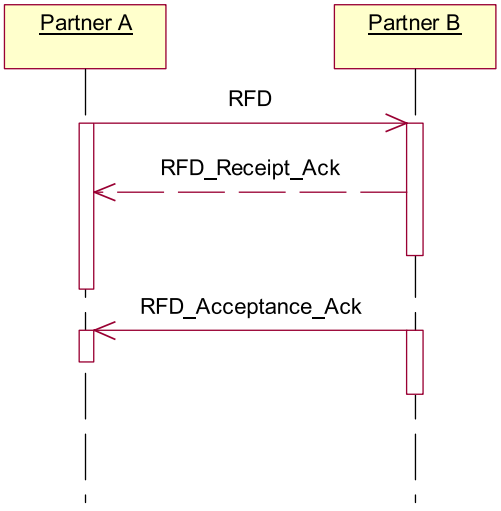


Fig. 9. Message sequence diagram of RFD primitive.

The following business constructs are based on the primitives previously discussed:

- RFD business construct (RFD primitive + Accept/Reject primitive + DS primitive)
- RFU business construct (RFU primitive + US primitive)
- RFI business construct (RFI primitive + IS primitive)
- US business construct (US primitive)
- IS business construct (IS primitive)

Based on these business constructs, collaborators can define any complex business scenario if they so desire. A standard representation for a business process modeling language, such as Business Process Execution Language (BPEL4WS) [8], can be used to represent CxP business constructs. Once represented by BPEL4WS, multiple business constructs can form a business scenario, which can be dynamically composed by using dynamic composition technology for Web services flow such as Web Services Outsourcing Manager (WSOM) [19,24].

3.2.2. Example: RFD business construct

A RFD business construct may contain one RFD primitive, one Accept/Reject primitive, and one DS primitive. The RFD micro-flow can be represented using BPEL4WS in Table 6. "process" in Table 6 stands for "business process", which is a standard way to describe business process in BPEL4WS.

In fact, the business collaboration or design collaboration patterns can be very complicated as they often involve multiple interactive messages based on the primitive protocols. Take design outsourcing for example. In Fig. 8, the right-hand side shows a design center, Product design team, which out-sources parts of the ThinkPad design to different design partners, e.g., to ABC for ASIC chip design, to XYZ for motherboard design, and to YAP for mechanical and electrical design. The left-hand side of Fig. 8 shows the design collaboration patterns between Product design team and ABC. The various messages flow between the two partners, starting from a request for design (RFD), followed by requests for updates (RFUs) by the acceptance of the RFD, by the submission of design, by the viewing of the design, by further RFUs and inter-

Table 4
Sample RFD message in CxP

```
<RDFNsId2:RFD
rdf:about='urn:RFD:SoundCard_test1@SoundCard_test1@SoundCard@Workstation@PDT@PDT'
RDFNsId2:description=''
RDFNsId2:identifier='SoundCard_test1'
RDFNsId2:status='Accepted'>
<RDFNsId2:requirement>
<RDFNsId2:DesignRequirement
rdf:about='urn:DesignRequirement:test1@SoundCard@Workstation@PDT@PDT'
RDFNsId1:cpuFrequency='500'
RDFNsId2:identifier='test1'
RDFNsId2:name='test1'
RDFNsId2:description='test1'>
<RDFNsId2:specification>
<rdf:Bag>
<rdf:li>
<RDFNsId2:Specification rdf:about='urn:Document:test@test1@SoundCard@Workstation@PDT@PDT'
RDFNsId2:identifier='test'
RDFNsId2:name='test'
RDFNsId2:description='d:\there'/>
</rdf:li>
<rdf:li>
<RDFNsId2:Specification rdf:about='urn:Document:@test1@SoundCard@Workstation@PDT@PDT'
RDFNsId2:description=''
RDFNsId2:name=''
RDFNsId2:identifier=''/>
</rdf:li>
</rdf:Bag>
</RDFNsId2:specification>
<RDFNsId2:reference
rdf:type='http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag'/>
<RDFNsId2:forTask rdf:resource='urn:Task:SoundCard@Workstation@PDT@PDT'/>
</RDFNsId2:DesignRequirement>
</RDFNsId2:requirement>
<RDFNsId2:transaction
rdf:resource='urn:Transaction:SoundCard_test1@SoundCard@Workstation@PDT@PDT'/>
<RDFNsId2:creationTime>Jan 16, 2003 5:10:17 PM</RDFNsId2:creationTime>
</RDFNsId2:RFD>
```

mediate reviews, finally concluded with the acceptance of design.

The sample design collaboration pattern shown in Fig. 8 demonstrates the flexibility and versatility of the RDF to support various data format required in collaboration message flow and document exchanges.

4. Related work

In today's Web services infrastructure [20], there is a lack of a uniform semantic representation for individual solution components. For example, WSDL concentrated on describing the basic information about a Web service; Some information about WSDL are published in Web services registries, namely, UDDI registry or WSIL documents, which are two different type of Web services registries. However, there is no place to describe capability information about a Web service, method signature mapping, and the like in current Web services related specifications. Moreover, there remains a set of open issues for incorporating WS-Security mechanisms within a project context, or business flow (e.g., BPEL4WS) context, and other solution components such as UDDI registry, SOAP invocation engine, and even WSDL documents. This is one of the rationales for forming WS-I Forum [18] in order to address the interopera-

Table 5
Example message for accept primitive of RFD

```
<RDFNsId2:AcceptanceAck
rdf:about='urn:AcceptanceAck:SoundCard_test1AcceptanceAck@
    SoundCard_test1@SoundCard@Workstation@PDT@PDT'
RDFNsId2:identifier='SoundCard_test1AcceptanceAck'
RDFNsId2:ack='Accept'
RDFNsId2:status='Sent'>
<RDFNsId2:transaction
rdf:resource='urn:Transaction:SoundCard_test1@SoundCard@Workstation@PDT@PDT'/>
<RDFNsId2:creationTime>Jan 16, 2003 5:38:38 PM</RDFNsId2:creationTime>
</RDFNsId2:AcceptanceAck>
```

bility issue among multiple standards specifications. Additional languages include Web Service Choreography Interface and others jointly defined by major e-business companies [1].

Most researchers in the field of Web services are realizing that semantic information are needed for effective Web services discovery [6,23], dynamic Web services composition as well as collaboration at run-time. Some semantic representation approaches have been proposed to address this issue. For example, (1) DAML-S is being proposed to describe more information about individual Web services for discovery and composition [5]; (2) Regular XML-based annotation languages have been defined to capture different types of semantic information. One example XML annotation, Web services relationship language (WSRL) [22], is proposed to capture the Web services relationships at different granularities, which we think will be an important facilitator in selecting and composing the right set of services that meets the customer's requirements. Additionally, a business requirement language, Business Process Outsourcing Language (BPOL) is proposed to capture the business requirements such as conceptual flow, preferences, business rules, relationship bindings, and event-action mappings for automating the Web services discovery and flow composition that matches customers' requirements [19,24]. (3) Organizational behaviors associated with an e-business solution refer to the semantic representations about the organizations, the on-going projects in an organization, tasks in a project, requirements and transactions in a task, additional annotation about any other related resources such as value-added services involved in an e-business solution.

For the first two, namely (1) and (2), as we mentioned earlier, we can find some example solutions to address the semantic representations for different aspects about individual Web services. For example, in traditional e-commerce environment, a workflow-based document routing language was proposed to address the inter-organizational collaboration [10]. It could be a very useful foundation for helping create intelligent documents. However, it is a regular XML-based description language which lacks the extensibility and schema-less feature. Moreover, there was no configurable business protocol creation framework to enable the business process based collaboration across multiple enterprises. The third is more systemic view of a semantic representation for building and managing a Web services based e-business solution. We think Resource Definition Framework (RDF) [13] could provide a uniform and efficient way to capture all three types of semantic representations. In this paper, the realization of Web services collaboration [20] is to create an extended business collaboration ontology, which is built on top of RDF [13], DAML-S [5], and other XML-based semantic representations, to effectively create and manage the Web services based extended business collaboration solutions.

Other existing human-based collaboration activities such as chat, session-sharing, white boarding and document shared-access [2] do not provide links to the B2B processes as they are centered only on human communication. In the solution framework outlined in the paper, we provide the means, technology and implementation for a deployment and operation of the human-assisted collaboration integrated in the context of managed business processes. These aspects alone cover a white space in the interaction and collaboration of enterprises in the new era of global commerce [21], outsourcing and cross-enterprise busi-

Table 6
RFD business construct in BPEL4WS

```
<process name="RFDmicroflow"
        targetNamespace="urn:samples:BusinessConstructs"
        xmlns:tns="urn:samples:BusinessConstructs"
        xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
  <partners>
        <partner name="RFDoriginator"
                serviceLinkType="tns:RFDoriginatingSLT"
                myRole="RFDoriginating"/>
        <partner name="RFDreceiver"
                serviceLinkType="tns:RFDreceivingSLT"
                myRole="RFDreceiving"/>
        <partner name="buyer"
        serviceLinkType="tns:buyingSLT"
        MyRole="buying"/>
  </partners>
  <variables>
        <variable name="RFDinvoke" messageType="tns:RFDinvoke"/>
     <variable name="RFDmsg" messageType="tns:RFDmsg"/>
        <variable name="RFD_Receipt_Ack" messageType="tns:RFD_Receipt_Ack"/>
        <variable name="Accept" messageType="tns:Accept"/>
        <variable name="DSinvoke" messageType="tns:DSinvoke"/>
        <variable name="DSmsg" messageType="tns:DSmsg"/>
        <variable name="DS_Receipt_Ack" messageType="tns:DS_Receipt_Ack"/>
  </variables>
   <correlationSets>
        <correlationSet name="POIdentifier" properties="POIdentifier"/>
        <correlationSet name="RFDIdentifier" properties="RFDIdentifier"/>
   </correlationSets>
  <sequence>
     <receive partner="buyer" portType="tns:buyerPT"
                        operation="purchase" variable="RFDinvoke"
                        createInstance="yes" name="ReceivePurchase">
                        <correlations>
                        <correlation set="POIdentifier" initiate="yes"/>
                </correlations>
        </receive>
        <invoke name="invokeRFDoriginator"
                partner="RFDoriginator" portType="tns:RFDoriginatorPT"
                operation="sendRFD" inputVariable="RFDinvoke" outputVariable="RFDmsg">
        </invoke>
        <invoke name="invokeRFDreceiver"
                partner="RFDreceiver" portType="tns:RFDreceiverPT"
                operation="receiveRFD" inputVariable="RFDmsg"
outputVariable="RFD_Receipt_Ack">
        </invoke>
        <invoke name="invokeRFD_Accept_Ack"
                partner="RFDreceiver" portType="tns:RFDreceiver"
                operation="sendRFDAccept"inputVariable="RFDmsg"
outputVariable="RFD_Accept_Ack">
        </invoke>
    <invoke name="invokeRFD_Accept_receive"
                partner="RFDoriginator" portType="tns:RFDoriginator"
                operation="receive_Accept" inputVariable="Accept">
```

Table 6 (*continued*)

```
        </invoke>
        <invoke name="invokeDS"
                    partner="RFDreceiver" portType="tns:RFDreceiver"
                    operation="submitDS" inputVariable="DSinvoke" outputVariable="DSmsg">
        </invoke>
    <invoke name="invokeDS_Receipt_Ack"
                    partner="RFDoriginator" portType="tns:RFDoriginator"
                    operation="receiveDS" inputVariable="DSmsg" outputVariable="DS_Receipt_Ack">
        </invoke>
    </sequence>
</process>
```

ness activities. The enablement of these extended interactions, collaborations and services, by leveraging web services technology, is an emerging area with the promise of reducing the overhead of implementation and by providing an ‘‘impedance-match’’ to allow companies with different systems, legacy applications, and various means of implementing business processes to participate in their respective business value nets with practical levels of investments in software and at competitive integration cost. These observations are based on our customer engagement experience in support of industry solutions.

The environment of interest in this paper encompasses business process flows that are deployed on different enterprise sites. Hence, the research focus of this paper is to find an efficient and effective approach to capture the business collaboration context (e.g., organizational structure, projects, tasks, requirements and the relationships among them) and configure customized business protocols to facilitate the information exchange among loosely coupled business processes.

The relationship among CxP, RosettaNet and BPEL4WS can be summarized as follows. CxP identifies the primitives for the collaborative ‘‘Partner Profile Processes (PIP)’’ (in the RosettaNet sense) [14] as well as the extendable hyperlinked data descriptions. CxP is immediately compatible at a high level with the RosettaNet PIP model. As we have illustrated in this paper, CxP can be restructured to be BPEL4WS compatible. The CxP Message data is extensible to support hyperlinked document types with RDF graph. They are used to compose collaborative business primitives such as the different variety of Request for Information (RFI) and Request for Updates (RFU).

One of the disadvantages of the HyperChain Manager presented in this paper is that it cannot directly process the platform or channel specific information such as CAD files, RosettaNet protocols, ebXML protocols, etc. However, the HyperChain Manager can route these activities to the right applications based on the business-annotated data carried in the CxP messages.

5. Conclusions and future work

In this paper, we have presented an on-demand business collaboration solution approach that supports a new model of integration of collaboration workplace with B2B collaborative process flow. A new breed of ontology-based information-exchange protocols, human–machine process primitives for design collaboration, middleware and complementary tool to support rapid B2B collaboration process design were also created to support the evolving domain of extended business collaboration paradigm.

The Collaborative exchange Protocols (CxP) stack of eBC supports elements of various granularities, which are configurable building blocks for creating adaptive solutions to achieve a business goal. What we have learned while developing the CxP enabling infrastructure is that modularizing and composing the reusable components in a flexible and extendable way was one of the major challenges. The traditional approach of using code template did not address the flexible configuration of new business protocols in the service oriented computing environment. However, the proposed eBC model and enabling framework brings human assistance aspect into different levels of a business process, i.e., at the CxP primitive level,

business construct level, and solution scenario level, for exception handling, escalation, and decision making. Web portal server is used to integrate people's activities into collaborative business processes.

Finally, we foresee the following issues that can be treated as future research topics in the field of extended business collaboration.

- Multiple variety of business scenarios can be simulated by supporting the coordination of multiple BPEL4WS sequencing fragments through an extension mechanism such as a BPEL++.
- Using eBC infrastructure and configurable business protocol framework to extend RosettaNet specification and enabling infrastructure to support flexible payload formats as well as to compose PIPs in a manageable way.
- Defining more industry specific ontology extensions for eBC ontology to create a rich business collaboration foundation.
- Creating a pluggable business scenario creation framework to reuse or extend the existing solution components to satisfy the changing customers' requirements.
- More work is required to define an adaptive integration action manager that can seamlessly integrate a new application into eBC environment by minimizing the code changes of the existing components in eBC infrastructure.
- Investigating the convergence of UML based model driven approach [12] and RDF based semantic approach for creating eBC solutions.
- Enabling eBC solution infrastructure in Autonomic [9] and Grid computing environment [7] for business process outsourcing, integration and collaboration [25].

Acknowledgements

We would like to thank Henry Chang and Jen-Yao Chung for their support of this research project. We also would like to thank Cristiane Hilkner and Pierre Darmon for their leadership in delivering an eBC prototype. The evolution of the on-demand business collaboration solutions underlying this work was not possible without the key contributions in design and implementation of dedicated outstanding scientists and engineers. They include Tian Chao, Shunxiang Yang, Jingming Xu, Yingnan Zuo, David Flaxer, Santosh Kumuran, Yiming Ye, Shang Guo, Yu Long, and numerous others. We also like to extend our thanks to Michael Silfen for his valuable business input in the preparation of this paper.

References

[1] S. Aissi, P. Malu, K. Srinivasan, E-business process modeling: the next big step, IEEE Computer 35 (5) (2002 May) 55–62.
[2] G. Bafoutsou, G. Mentzas, A comparative analysis of web-based collaborative systems, 12th International Workshop on Database and Expert Systems Applications, 2001, vol. 3–7, IEEE Computer Society, USA, 2001 (Sept.), pp. 496–500.
[3] D. Brickley, R.V. Guha, RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema/ (2003 Sept.).
[4] Cadence White Paper on Design Chains, http://www.cadence.com/.
[5] DAML-S: DAML Services, http://www.daml.org/services/.
[6] ETTK, Emerging Technologies Toolkit, IBM alphaWorks, http://www.alphaworks.ibm.com/tech/ettk.
[7] I. Foster, C. Kesselman, J.M. Nick, S. Tuecke, Grid services for distributed system integration, Computer 35 (6) (2002) 37–46.
[8] IBM, Microsoft, SAP, and Siebel Systems, Business Process Execution Language (BPEL4WS), http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.
[9] J.O. Kephart, D.M. Chess, The vision of autonomic computing, IEEE Computer 36 (1) (2003) 41–50.
[10] A. Kumar, J.L. Zhao, Workflow support for electronic commerce applications, Decision Support Systems 32 (2002) 265–278.
[11] OASIS eXtensible Access Control Markup Language, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml (2003).
[12] OMG Model Driven Architecture, http://www.omg.org/mda/.
[13] RDF: Resource Description Framework, http://www.w3.org/RDF/.
[14] RosettaNet, http://www.rosettanet.org.
[15] R. Schmelzer, et al., XML and Web Services Unleashed, SAMS Publishing, Indiana, USA, 2002.
[16] WebSphere Business Integration Connection, http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/businessint.
[17] WebSphere software platform, http://www-3.ibm.com/software/info1/websphere/index.jsp.
[18] WS-I, Web Services Interoperability Organization, http://www.ws-i.org.
[19] WSOM, Web Services Outsourcing Manager, http://www.alphaworks.ibm.com/tech/wsom/, IBM developerWorks USA (2002).
[20] L.-J. Zhang, M. Jeckle, The next big thing: web services

collaboration, Proceedings of the 2003 International Conference on Web Services – Europe (ICWS-Europe'03), LNCS 2853, Spring-Bergle, Sept. 23–24, 2003, Erfurt, Germany.
[21] L.-J. Zhang, H. Chang, T. Chao, J.-Y. Chung, Z. Tian, J. Xu, Y. Zuo, S. Yang, Q. Ao, Web services hub framework for e-sourcing, in: IEEE Conference on System, Man, and Cybernetics (SMC'02) vol. 6, IEEE, USA, 2002, pp. 163–168.
[22] L.-J. Zhang, H. Chang, T. Chao, Web services relationships binding for dynamic e-business integration, International Conference on Internet Computing (IC'02), Las Vegas, CSREA Press, USA, 2002, pp. 561–567.
[23] L.-J. Zhang, T. Chao, H. Chang, J.-Y. Chung, XML-based Advanced UDDI Search Mechanism for B2B Integration, Electronic Commerce Research Journal, (2003) 25–42.
[24] L.-J. Zhang, B. Li, T. Chao, H. Chang, On Demand Web Services-Based Business Process Composition, IEEE International Conference on System, Man, and Cybernetics (SMC'03), pp. 4057–4064.
[25] L.-J. Zhang, Q. Zhou, J.-Y. Chung, Developing Grid Computing Applications, IBM DeveloperWorks Journal, 2003 (May), pp. 10–15.

Dr. John Sayah is currently the Program Director of Emerging Business Opportunities Strategy for the Industry Solutions in IBM Software group. His main focus is on emerging business opportunity initiatives and linkages of research and industry teams for driving next generation solution offerings. Dr. Sayah has held several management and technical lead assignments at IBM covering a wide spectrum of application and industry areas including business-to-business and business integration, internet service provider management, and computer-aided-design applications. John Sayah received a PhD in Computer Engineering from the University of Wisconsin-Madison, an MSc in Electrical and Electronics Engineering from the University of London-UK, and a Licence D'enseignement in Physics from the Lebanese University in Beirut-Lebanon.

Dr. Liang-Jie (LJ) Zhang joined IBM China Research Lab in 1996 and is currently a Research Staff Member at IBM T.J. Watson Research Center. He is part of the e-Business solutions research team with a focus on collaborative business process integration and management innovations. He is actively creating novel business process integration solutions by leveraging and enhancing Web Services and Grid Computing technologies. Before this position, LJ was the lead inventor of IBM HotVideo technology and a key architecture board member of IBM HotMedia Product. Dr. Zhang has 5 issued patents, 25 filed patent applications, and about 70 published papers. He is an IEEE Senior Member and the Chair of the IEEE Computer Society Technical Community for Services Computing (TCSC). He is the General Chair of the 2004 IEEE International Conference on Web Services (ICWS 2004) and the General Co-chair of the 2004 IEEE Conference on E-Commerce Technology (CEC 2004). Currently, he is the Editor-in-Chief of the International Journal of Web Services Research (JWSR). Liang-Jie received a BS in EE at Xidian University in 1990 and then at Xi'an Jiaotong University an MS in EE in 1992 and a PhD in Pattern Recognition and Intelligent Control in 1996 at Tsinghua University.

2009 IEEE International Conference on Web Services

CCOA: Cloud Computing Open Architecture

Liang-Jie Zhang and Qun Zhou
IBM T.J. Watson Research Center, New York, USA
E-mail: {zhanglj, qzhou}@us.ibm.com

Abstract

Cloud Computing is evolving as a key computing platform for sharing resources that include infrastructures, software, applications, and business processes. Virtualization is a core technology for enabling cloud resource sharing. However, most existing Cloud Computing platforms have not formally adopted the service-oriented architecture (SOA) that would make them more flexible, extensible, and reusable. By bridging the power of SOA and virtualization in the context of Cloud Computing ecosystem, this paper presents seven architectural principles and derives ten interconnected architectural modules to form a reusable and customizable Cloud Computing Open Architecture (CCOA). Two case studies on Infrastructure and Business Cloud are used to deliver business and practical value of infrastructure and business process provisioning services over the Internet. We also present some potential value-added services of the proposed CCOA to guide strategic planning and other consulting practices of Cloud Computing.

1. Introduction

As a key service delivery platform in the field of service computing [12], Cloud Computing provides environments to enable resource sharing in terms of scalable infrastructures, middleware and application development platforms, and value-added business applications. The operation models may include pay-as-go utility models, free infrastructure services with value-added platform services, fee-based infrastructure services with value-added application services, or free services for vendors but sharing of revenues generated from consumers.

As summarized in [1], typically there are four types of resources that can be provisioned and consumed over the Internet. They can be shared among users by leveraging economy of scale. Provisioning is a way of sharing resources with requesters over the network. One of the major objectives of Cloud Computing is to leverage Internet or Intranet to provision resources to users.

The first type of resources is infrastructure resources, which include computing power, storage, and machine provisioning. For example, Amazon EC2 provides web service interface to easily request and configure capacity online [2]. Xdrive Box service provides online storage to users [3]. Microsoft SkyDrive provides free storage service, with an integrated offline and online model that keeps privacy-related files on hard drives, and enables people to access those files remotely [4]. In the area of computing power sharing, the Grid computing initiative has taken it as its major focus to use clustering and parallel computing technologies to share computing power with others, based on task scheduling when computers are idle.

The second type of resources in Cloud Computing is software resources including middleware and development resources. The middleware consist of cloud-centric operating systems, application servers, databases, and others. The development resources comprehend design platforms, development tools, testing tools, deployment tools, and open sources-based reference projects.

The third type of resources in Cloud Computing is application resources. The leading companies in the information industry are gradually moving applications and related data to the Internet. Software applications are delivered through Software As A Service (SaaS) model or mashups of value-added applications. For example, Google has used Cloud Computing platform to offer Web applications for communication and collaboration [19]. Google docs move productivity applications to the Web, gradually replacing heavy-weighted desktop applications. Therefore, developing a reusable and customizable Cloud Computing Open Architecture for enabling application development environment is a key to success of application sharing over the Internet.

The fourth type of resources in Cloud Computing is business processes. Some applications can be exposed as utilities, namely loosely coupled sub-processes or tasks inside customers' business processes. Business process sharing is the business-driven application outsourcing that supports reuse, composition, and provisioning.

This paper is organized as follows. The first part of Section 2 discusses the two key enabling technologies, Virtualization and SOA, for articulating the value of creating an open architecture for Cloud Computing. The rest of Section 2 presents a Cloud Computing Open Architecture (CCOA) and its seven principles for building extensible and flexible Cloud Computing systems and applications. Section 3 describes two case studies on Infrastructure Cloud and Business Cloud, to illustrate infrastructure service provisioning and business process as a service. Some value-added offerings based on the proposed CCOA are also depicted. Section 5 discusses

978-0-7695-3709-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICWS.2009.144

related work and future directions. Conclusions are drawn in the end of this paper.

2. Cloud Computing Open Architecture

Currently, there is no standard definition or specification for Cloud Computing. It may take some time to define the key characteristics of Cloud Computing based on practices in the field. Cloud Computing involves a set of key technologies to address resource sharing based on business requirements. Based on our practices in the areas of service provisioning and solution design, we think the following two key enabling technologies could play very important roles in this revolutionary phase: virtualization technology and Service-Oriented Architecture (SOA).

The virtualization technology handles how images of the operating systems, middleware, and applications are pro-created and allocated to the right physical machines or a slice of a server stack. The images could be moved around and put into production environment on demand. On the other hand, virtualization technology can also help reuse licenses of operating systems, middleware, or software applications, once a subscriber releases his/her service from the Cloud Computing platform.

The SOA is the evolution of a system or software architecture for addressing componentization, reusability extensibility, and flexibility. In order to construct scalable Cloud Computing platforms, we need to leverage SOA to build reusable components, standard-based interfaces, and extensible solution architectures.

Creating a so-called Cloud Computing platform is easy as long as it can enable sharing of at least one of the resources. However, building a unified, scalable and reusable Cloud Computing architecture to support sharing of all types of resources still faces challenges in the areas of technology breakthrough and best industry practices.

2.1 "OSI" Model for Cloud Computing

We have identified the following three objectives to help address the above challenge of defining a good open architecture for Cloud Computing.

The first objective is to articulate a reusable way of creating scalable and configurable provisioning platform for Cloud Computing. This paper brings together the power of Service-Oriented Architecture (SOA) and virtualization to deliver business and practical value to emerging software applications, hardware, and business process provisioning services over the Internet in the context of Cloud Computing.

The second objective is to propose a set of common and shared services for building Cloud Computing platforms, to provide business services or other cloud offerings to its enterprise consumer users in a unified approach.

The third objective is to maximize the potential business value of Cloud Computing based on an extensible IT infrastructure and management system. This will lead to the value added services of business cloud, through monetization of the combined power of SOA and Cloud Computing.

As we know, OSI standards for Open System Interface [6] which has encountered some challenges to realize its value in the context of generic open systems. In this paper, we try to limit the scope of the open system to a specialized domain, and leverage service-oriented thinking to help modularize open architecture for Cloud Computing. In the following section, we present a Cloud Computing Open Architecture based on seven principles.

2.2 Seven Principles of Cloud Computing Architecture

In this Cloud Computing Open Architecture, we propose an integrated co-innovation and co-production framework to get cloud vendors, cloud partners, and cloud clients to work together based on seven principles, which are used to define ten major architectural modules and their relationships shown in Figure 1. The presented Cloud Computing Open Architecture covers cloud ecosystem enablement, cloud infrastructure and its management, service-orientation, cloud core on provisioning and subscription, compostable cloud offerings, cloud information architecture and management, and cloud quality analytics. This is a logical and modularized separation, which helps isolate concerns of details of each module during the design process. Since the connections between the identified key architectural principles for Cloud Computing are quite complex, the information exchanges are going through the Cloud Information Architecture and Cloud Ecosystem Management. In the rest of the section, we will introduce the details of each principle illustrated in Figure 1.

Principle 1: Integrated Ecosystem Management for Cloud

An architecture must support the management of the ecosystem of Cloud Computing. This ecosystem includes all involved services and solutions vendors, partners, and end users to provide or consumer shared resources in the Cloud Computing environment. Cloud vendors expose its interaction interfaces of its internal operations and product development capability to the cloud. Cloud partners provide components to cloud vendors or serve as agents to provide value-added services to the cloud clients. Cloud clients are users of the cloud services that offer business goal driven resource sharing.

From an architecture design perspective, cloud vendor dashboard provides an integrated view of interaction with vendors' frontend and backend operations. For example, in the frontend, marketing and business development activities, services delivery portals, and customer support

can be enabled in the cloud vendor's dashboard. In the backend, solution design and development activities, or hosting environment are used to support the frontend's operations.

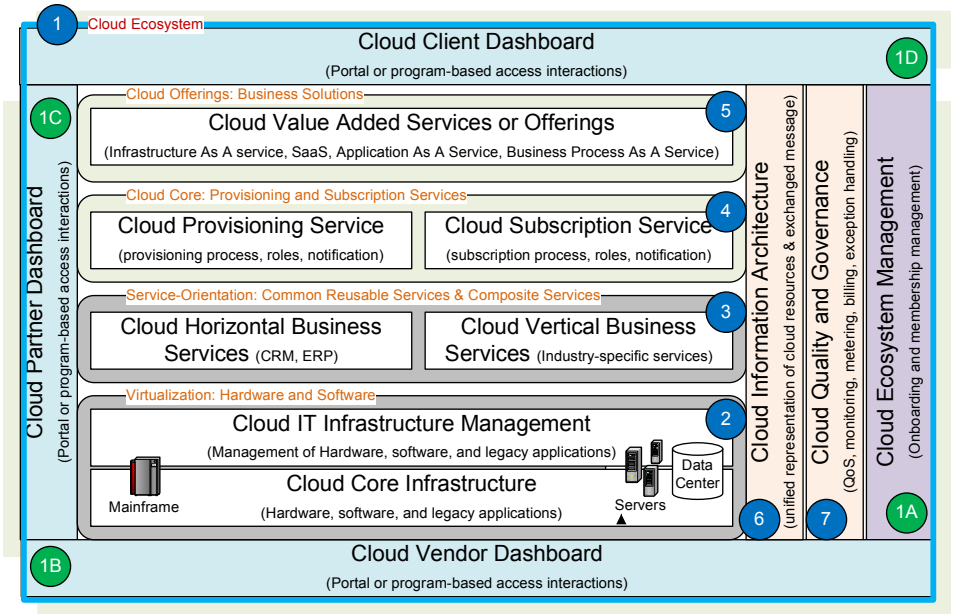


Figure 1. Cloud Computing Open Architecture Overview Diagram

Since most of the cloud vendors do not work alone anymore, they need to collaborate with their partners [7] in the value chain of Cloud Computing environment. In this regard, a partner dashboard is needed for the participating partners to interact with the cloud vendors and clients. For example, if the cloud partners serve as component suppliers for the cloud vendors, architectural building blocks for interacting with vendors and collaboration policy manager are keys to the value chain integration.

Clients or end users of Cloud Computing can be grouped into two classes: enterprise and consumer users. The cloud client dashboard provides a focal point for all kinds of users to interact with Cloud Computing services or offerings. This focal point provides a unified framework for users to consume cloud services via multiple channels such as Web portal, program-based business to business collaboration channel, or phone-based customer representative channel. There are opportunities to explore a converging software and services architecture for enterprise and consumer users based on various pricing strategies, security enablement, and other features of software and services. Since enterprise users or consumer users are co-existing role players in the service ecosystem, an enterprise user may have multiple consumer users. In the end, they are just consumers of Cloud Computing resources at different levels.

Putting all those dashboards together, the Cloud Computing ecosystem management layer (1A) provides an integrated on-boarding process and common utilities to support the seamless collaboration and message exchanges among cloud vendors, partners, and clients. For example, the onboard progress covers the registration of business entities and users. The business entities include cloud vendors, cloud partners, and enterprise cloud clients. The user entities are end users within a certain business entity (e.g. an employee of a company, or a member of a registered community like a social network), or consumer users in the open Internet space.

Principle 2: Virtualization for Cloud Infrastructure

There are two basic approaches for enabling virtualization in the Cloud Computing environment. The first approach is hardware virtualization that is to manage hardware equipments in plug-and-play mode. Hardware equipments can be added or removed without affecting the normal operations of other equipments in the system. Of course, performance or storage spaces may be dynamically changed due to those add and remove actions.

The second approach is software virtualization, i.e., to use software image management or software code virtualization technology to enable software sharing. Specifically, software images can be created based on the degree of reusability of a set of software systems including operating system, middleware, and applications. The other software virtualization technology is dynamic code assembly and execution. In this case, there are no software images. Code elements will be dynamically copied from repositories and pasted in right places based

on business logic. For instance, in an Internet application, some JavaScript code elements can be dynamically retrieved and inserted into a right Ajax package to create new functions or features for a Web client. By the same token, server side programs can be dynamically assembled and executed based on the composition of reusable code elements and just-in-time complier technologies. With the development of multi-core technology and parallel programming, this dynamic code assembly and execution technology will have great advantages moving forward. Especially, it will get rid of the requirements of huge storage spaces for software images for middleware and development tools. But in today's environment, both software virtualization technologies can co-exist and support each other based on usage scenarios.

In short, the Cloud IT Infrastructure Management module covers software image management, hardware virtualization, and legacy application packaging.

The target resources being managed by the Cloud IT Infrastructure Management module is the Cloud Core Infrastructure, which comprises all supporting hardware, software, and legacy applications for operating a Cloud Computing environment. For example, hardware may include mainframe, distributed servers or clusters, and data storages. Software may include database packages and related middleware. Legacy applications may involve home-grown applications or ISV applications that are part of the infrastructure.

It is noted that this virtualization principle in the Cloud Computing Open Architecture is an extension of the operational system layer in the SOA Solution Stack (a.k.a. SOA Reference Architecture) [8] in the context of Cloud Computing enablement. For example, a rack manager can be leveraged to handle wire hardware systems. The interface of dynamically assembling IT resources enables infrastructure resource provisioning, and can be used to host or deploy business applications that leverages distributed cloud infrastructure resources.

Principle 3: Service-Orientation for Common Reusable Services

As introduced before, in addition to the virtualization characteristic, service-orientation is another driving force to enable Cloud Computing to further realize the business value from asset reusability, composite applications, and mashup services. There are two major types of common reusable services: Cloud Horizontal and Vertical Business Services.

The Cloud Horizontal Business Services consist of various platform services that hide the complexities of middleware, database, and tools. In addition to offering middleware or development tools as services in the Cloud Computing environment, some common utilities such as on-boarding, provisioning, monitoring, billing tools, or cross-industry services like Customer Relationship Management (CRM) and Enterprise Resource Planning (ERP).

The Cloud Vertical Business Services include all domain specific or industry-specific utility services. Examples are shipping and payment services.

Both common reusable services in CCOA can be reused to enable Cloud core's provisioning and subscription services, as well as to build cloud offerings such as Infrastructure As A Service, SaaS, Application As A service, Business Process As A Service, which will be further described in Principle 5.

Principle 4: Extensible Provisioning and Subscription for Cloud

Extensible service provisioning is the unique feature of a Cloud Computing system. Without extensibility, the provisioning part of the Cloud Computing architecture can only support a certain type of resource sharing. This implies that the service provisioning architecture for free-use users and paying users are the same. Both types of users can be service providers or consumers from time to time. From service consumers' perspective, they are interested in how to easily access services based on their own business logics and goals. From service providers' perspective, three levels of service provisioning described in Section 1 will be the target offerings.

The Cloud Core shown in Figure 1 includes a set of provisioning and subscription services. This addresses a key question of how to handle service providers' provisioning process and service consumers' subscription process in the Cloud Computing architecture. We have categorized Cloud Core into Cloud Provisioning and Subscription Service. The key architectural elements of Cloud Provisioning Service include provisioning process, role definitions, and notification framework. Cloud Subscription Service involves subscription process, role definitions, and notification framework. The role defining framework and attributes as well as notification framework can be shared for both provisioning and subscription services.

Principle 5: Configurable Enablement for Cloud Offerings

Cloud offerings are the final products or services that are provisioned by the Cloud Computing platform. Since all cloud offerings should address certain business goals, cloud offerings are also known as cloud business solutions. In alignment with the categorization of resource sharing described in Section 1, CCOA defines its offering aspects at the following four levels: infrastructure as a service, software as a service (SaaS), application as a service, and business process as a service.

Leveraging the SOA reference architecture's extensibility and configurability, the proposed CCOA pursues configurable enablement of Cloud Computing platforms and services. The modularized cloud ecosystem management, virtualization, service-orientation, and cloud core have formed a solid foundation to ensure a computing platform to be configurable, compostable, and manageable.

In the category of infrastructure as a service, CCOA leverages the cloud core to support provisioning and subscription of the virtualized IT infrastructure resources by exploiting some common reusable services and cloud ecosystem management capabilities. Some example cloud offerings are storage cloud and infrastructure cloud.

Most cloud offerings are delivered or accessed through Web browsers. For instance, we can use web browser to upload photos, audio files, video files, and other documents to a storage cloud. In all other cloud offerings, Web interfaces have been proven effective channels as well to enable cloud clients to interact with cloud partners and vendors in the lifecycle of service delivery.

In the area of software as a service, lots of success stories demonstrate common reusable services such as CRM as a service (e.g. Saleforce.com), payment as a service (e.g. eBay's PayPal), and shipping as a service by leveraging the cloud core to manage the provisioning and subscription. It is emphasized that lots of value added services can be built on top of the common reusable services in CCOA before they are provisioned to cloud clients as cloud offerings. Therefore, composite applications or service composition technologies in SOA could play an important role in building value added cloud services in the context of CCOA.

In the application as a service area, lots of standalone applications or network-based applications can be provisioned as cloud services for sharing in the Cloud Computing value chain. For example, Web-based development tools could be good cloud offerings for the application development community, to design and develop applications for cloud or standalone applications by leveraging the Cloud Computing's development resources, such as enterprise modeling software, business process modeling software, architecture modeling software, application development software, and related asset repositories. Specifically, in the service-oriented solution design space, offering SOMA-ME [13] service as a design in the Cloud Computing environment is a natural way to empower mass community members of application design and development. There is no absolute boundary between software as a service and application as a service. In CCOA, an application is the integration of a set of software packages based on certain business logics and goals.

In the Cloud Computing environment, business process as a service is a new model for sharing best practices and business processes among cloud clients and partners in the value chain. An example business process as a service offering in the Cloud Computing environment is software testing, which is a very important business process in the lifecycle of software development. Since software systems or applications need to be deployed and run on different operating systems, middleware environments, or different versions or configurations of them, it is very hard for individual developers or companies to set up a testing "factory" to address those testing concerns in the area of application services. Upgrading or maintaining testing environment involves the migration of hardware, software, and testing knowledge repositories. They are the two major expenses in addition to the testing engineers who manipulate the environment to conduct the testing.

Principle 6: Unified Information Representation and Exchange Framework

Information representation and message exchange of Cloud Computing resources are very important to enable the collaborative and effective features of Cloud Computing. In CCOA, Cloud Computing resources include all business entities (e.g. cloud clients, partners, and vendors) and the supporting resources such as virtualization related modules, service-orientation related modules, cloud core, and cloud offerings. For example, in cloud ecosystem management, country, site, and organization are associated with the business entities in the Cloud Computing. Role players are used to define their functions in the dynamics of the Cloud Computing ecosystem. Project, task, documents, transactions, business process, reference links, annotation, and events are potential resources for supporting collaboration among various business entities in the Cloud Computing environment.

In CCOA, the cloud information architecture module enables representation of those cloud entities (business entities and supporting resources) in a unified Cloud Computing entity description framework. For example, technologies such as Resource Description Framework (RDF), Web Services Resource Framework (WSRF), and XML are candidates for implementing this unified framework. The messages exchanged between cloud entities form message exchange patterns. The message format and message exchange patterns can be reused to support various business scenarios. The message routing and exchange protocols as well as message transformation capability form a foundation for cloud information architecture.

Just like blood in human bodies, the cloud information architecture uses its information "blood" to form "blood stream" to get all various modules to communicate with each other in an effective way in CCOA.

Principle 7: Cloud Quality and Governance

The last and most important module in CCOA is the Cloud Quality and Governance shown in Figure 1. This module is responsible for the identification and definition of quality indicators for Cloud Computing environment and a set of normative guidance to govern the design, deployment, operation, and management of the cloud offerings.

From quality indicators' perspective, Quality of Services (QoS) parameters can be directly used to define cloud entities' reliability, response time, security, and integrity. The integrity can be checked through traceability enablement and compliance validation. Security is a very important aspect of the cloud quality. Only authorized business entities or users can access the right Cloud Computing resources. Access control, privacy, and protection of the cloud entities form the aspects of the trust and security foundation in CCOA.

From the governance perspective, lots of best practices from SOA governance can be borrowed to enable Cloud Computing environment and services offerings. For example, in order to launch a cloud initiative, a center of excellence or competence can be formed to better communicate and coordinate between business leaders and technical teams. In the project execution phase, monitoring, metering, billing, and exception handling are to be enabled and coordinated across line of businesses which produce, operate, and manage various modules of the cloud offerings.

3. Case Studies of CCOA

There are multiple scenarios of using the proposed CCOA, and no particular constraints with the design of any specific portions of the CCOA. For example, we can use CCOA to do high-level strategic planning for an enterprise to execute Cloud transformation initiative. We can also use CCOA to build an infrastructure stack to reduce operating cost. By the same token, we can use CCOA to guide the development or deployment practices of a Cloud Computing application. We can also create new value-added Cloud Computing applications based on the existing assets in a systematic way by following the principles of CCOA.

For an inter-connected Cloud Computing scenario, CCOA can be used as the architectural foundation to guide the design, development, deployment, and management of collaborative service delivery in the cloud value chain. From the methodologies' perspective, we can use the bottom-up approach to identify capabilities and provisioning of infrastructure. We can also use top-down approach to create cloud offerings and leverage or create cloud infrastructure to support the offerings.

In this section, we present two example cloud offerings based on CCOA. The first case is about Infrastructure Cloud, which is an example infrastructure as a service. The second case studies Business Cloud, which is an example of business process as a service in Cloud Computing environment.

3.1 Infrastructure Cloud

In this case study, we build our own private cloud to enable computing resource sharing. The usage scenario is to offer service requesters a pay-to-use model to provide servers with a selected set of preinstalled software packages to shorten time to production based on request submissions.

We use the 7 principles to instantiate CCOA for an Infrastructure Cloud to supply Server Provisioning Offering to cloud clients. Within the cloud ecosystem (Module 1), the Cloud Ecosystem Management module (1A) provides Web services user account management. Cloud Vendor Dashboard (1C) is responsible for handling the sign in process and provides a Web portal for cloud vendors to access the cloud ecosystem. The Cloud Partner Dashboard module (1C) in Figure 2 is not used in this case. The reason is that all virtualization capability of hardware and software is provided purely by cloud vendors in house. No cloud partners are involved in this Infrastructure Cloud. Cloud Client Dashboard supports sign-in process and a Web-based portal for cloud clients to submit order request and access related information on the provisioned servers. WebSphere Portal Server in Module 2 has been used to deploy UI related portlets.

In the virtualization module (Module 2), the Cloud Core Infrastructure includes a large number of servers, data centers, and supporting software packages such as WebSphere Application Server (WAS), DB2 database, and LDAP. In this case study, we use WebSpere Process Server to support BPEL processes and WebSphere Portal Server to support portlets. LDAP is used to manage users and access control information. Cloud Infrastructure Management is enabled by Tivoli Enterprise Portal Server and Tivoli Provisioning Manager, to handle the hardware virtualization and provisioning in this Infrastructure Cloud.

In the Service-Orientation module (Module 3), we have identified and defined a set of common reusable services based on service-oriented thinking. Cloud Horizontal Business Services contain cloud order management service, resource management service, Email notification service, and workflow management service. Here they are all implemented as Web services. The cloud vertical business services of module 3 are not used in this case study, since this infrastructure provisioning offering cloud does not leverage or provide domain-specific or industry-specific business services.

In the Cloud Core module (Module 4), Cloud Provisioning Services are supported by a set of business processes such as Service Provisioning (ServiceP), Service De-provisioning (ServiceDeP), and Contract Change including Contract Extension and Contract Termination. Cloud Provisioning Services are mainly used by Cloud Vendors to selectively share cloud resources. Cloud Subscription Service is enabled by Offering Creation and Order Creation business processes.

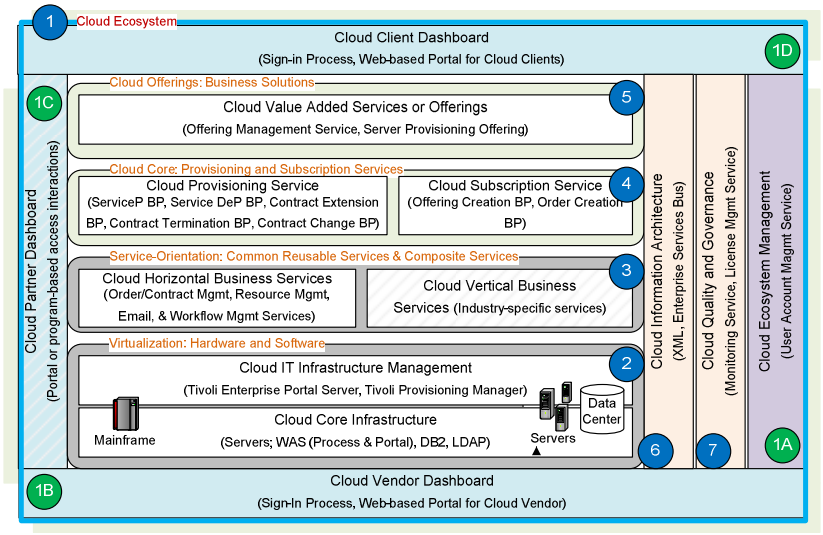


Figure 2. Instantiated Architecture for Infrastructure Cloud

In the Cloud Offering module (Module 5), the Infrastructure Cloud only provides Server Provisioning Offering to cloud clients. More value-added services can be enabled and offered in this Infrastructure Cloud environment without changing the basic modules of its architecture. For example, storage cloud offering can be created for the cloud vendor to share its data centers with cloud clients. All the creation and provisioning businesses of cloud value-added services are managed by Offering Management Service, which is realized in a Web service in this case study.

In the Cloud Information Architecture module (Module 6), we define data structures for subscription orders, contracts, SLAs, project information, and business scenarios. Those data structures and data are stored in DB2 database in the Virtualization module (Module 2). We also define data structures for users and groups. In the Virtualization module, LDAP is used to capture the user related information and access control. Enterprise Services Bus (ESB) technology is used to implement message routing and transformation in the cloud information architecture module.

In the Cloud Quality and Governance module (Module 7), service-level agreements, contracts, and resource statistics are defined based on the data structures defined in the Cloud Information Architecture module. For example, the start and end time of provisioning or de-provisioning services are captured in this module to ensure that services are delivered according to contracts. In the case of exceptions, we can also use this module to track where they come from and to provide solutions. From the enablement perspective, we use Tivoli Enterprise Portal Server to monitor the service operations in the Virtualization module (Module 2). Meanwhile, Tivoli Provisioning Manager can schedule and monitor all provisioning tasks. In the governance aspect, organizations like the Center of Excellence on Cloud Computing and Cloud Operations Team are created to support its overall strategic planning, architecture design and review, alignment, and coordination. The overall productivity, reusability, cost-effectiveness are also identified and reflected in the design of the Cloud solution architecture based on CCOA's seven principles. For example, some policies on when to use Web services and where to allocate machines are also part of the governance process.

As an example, we use the Cloud Vendor Dashboard to monitor the usages and resource statistics which are granted by the Cloud Quality and Governance module. Figure 3 illustrates the statistic data chart of the Cloud Computing resources. This usage diagram shows how many resources are available for cloud clients to subscribe. The resources are categorized based on server types and operating systems (e.g. Windows and Linux). In this case, there are Xen-VM, xSeries, Xen Cluster, VMware, DynamicP6, and DynamicP. We have two operating systems (Windows and Linux) for provisioning. As shown in Figure 3, there are one subscribed Xen-VM machine with Linux and one Xen-VM for subscription. There are 17 xSeries for subscription. There are 13 subscribed Xen Cluster machines with Linux and 9 machines for subscription. There are one subscribed

VMware machine with Windows and one VMware machine for subscription. There are 1 subscribed DynamicP6 with Linux and 5 machines for subscription. There are 4 DynamicP servers for cloud clients to subscribe.

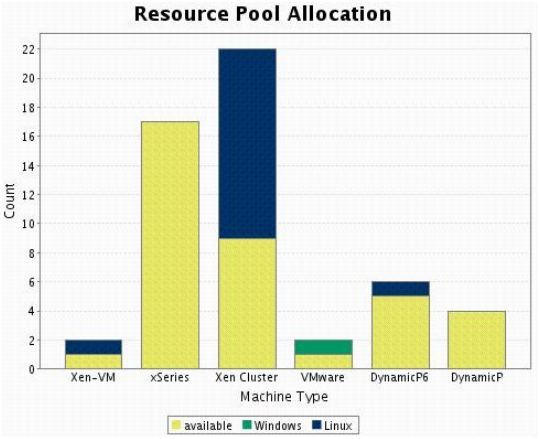


Figure 3. Cloud Resource Statistics

Figure 4 displays a screen of order creation in the Cloud Client Dashboard. Cloud clients can create orders after reading and accepting the license agreement which is maintained in the Cloud Quality and Governance module. The cloud user can pick available resources and their configurations such as operating system (OS), server type, number of CPUs, size of memory (GB), CPU sped (MHz), storage size (GB), and quantity shown in Figure 4.

The corresponding number of available resources is also shown in the dashboard based on the selection of the configurations. There are two categories of server offerings: standard and clustered. For clustered offerings, the cloud users can choose number of nodes in the order request. Once the configuration selection process is done, the order can be added to the shopping cart. The subsequent screens in the dashboard enable users to pick middleware and view the price information.

The dashboard also allows users to select team member for managing or using those resources. Once an order is submitted, the project managers or administrators of the Cloud Vendor Dashboard can review, approve, reject or delete it. After the order is approved, the user can view its status. All business logics and screen flows are implemented as BPEL processes and portlets. The data structure of an order in Figure 4 is defined in the Cloud Information Architecture module (Module 6) shown in Figure 2.

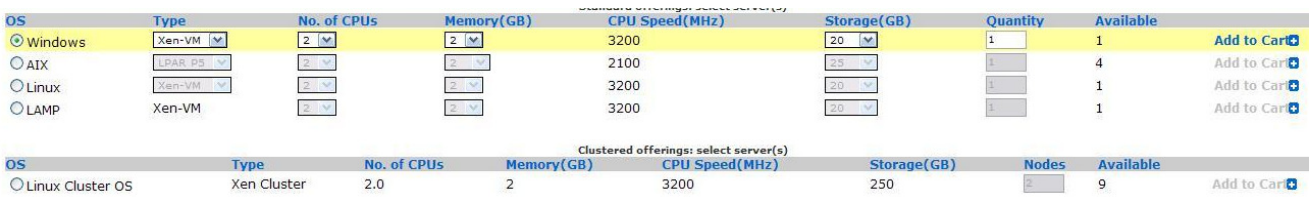


Figure 4. An Example Screen of the Cloud Client Dashboard

3.2 Business Cloud

Business Cloud covers all scenarios of business process as a service in the Cloud Computing environment. CCOA can be used to support the cloud offerings of business cloud. In this section, we employ a public cloud to demonstrate the usage of CCOA to enable business cloud offerings. In order to make a business process smarter, human intelligence can be introduced into it as a sub-process, human task, or decision making within a task.

As illustrated in Figure 5, a company called Cloud Publishing Business (CPB) would like to exploit the Cloud Computing platform to help extend its copyright protection business process that includes three major sub-processes. The first sub-process is Automatic Checking that triggers a third-party's plagiarism detection tool to generate an alert with a summary report when a new manuscript is submitted. A lot of scenarios need human intelligence to analyze the summary report by validating or correcting the findings and regenerate a report. For example, a published paper in a conference proceedings may be enhanced and submitted to a journal for consideration as long as this paper has at least 30% new contents. Or this paper may be enhanced based on his or her own blog entries in the social networking space. In those cases, human beings can perform a more effective job than computers in terms of validation, correction or enhancement of a manuscript. This is the second sub-process named Intelligent Validation and Analysis.

In this business cloud case study, we assume CPB has been convinced to execute this Intelligent Validation and Analysis sub-process in a scalable way. Then the challenge becomes how to quickly access qualified human beings to conduct a quality-guaranteed validation and analysis service. As an example, CPB leverages workforce marketplace such as Amazon Mechanical Turk (AMT) [11] to implement this sub-process. We use CCOA to separate the concerns of business execution and

supporting infrastructure. As shown in Figure 5, the workforce marketplace is enabled by Amazon Mechanical Turk which implements key modules of CCOA to support cloud-based publishing business. What CPB needs to do is to leverage Mechanical Turk APIs in its Cloud Client Dashboard to build its own CPB Portal which can co-exist with Amazon's client portal. From the business model perspective, CPB can create Copyright Detection Service to be used internally or as a value-added cloud offering for other publishers or authors to subscribe. The CPB Portal provides interfaces for CPB's customers to submit requirements and job descriptions. CPB launches its overall Copyright Detection business process and generates the initial summary report after the first sub-process completes. Then CPB Portal creates corresponding Human Intelligence Tasks (TITs) based on Amazon's Mechanical Turk APIs or Web services.

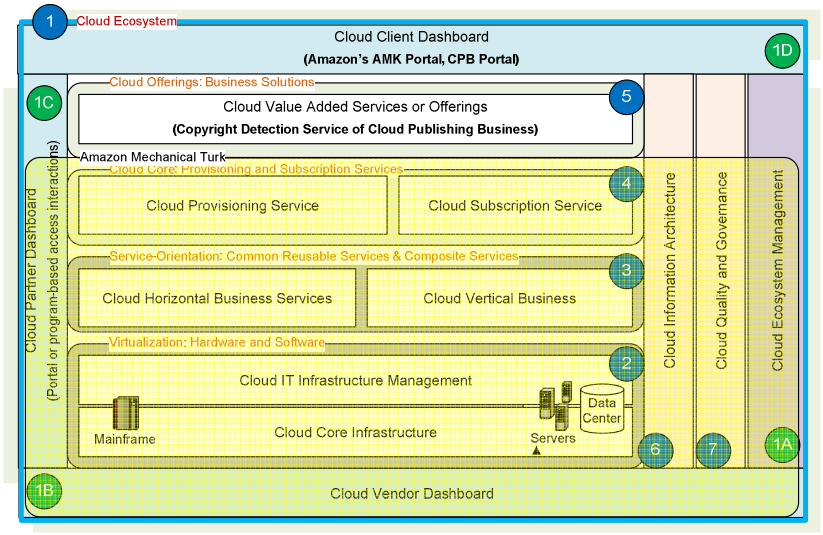


Figure 5. Architecture for Cloud Publishing Business

Theoretically, millions of HITs and qualification requirements can be automatically created in CPB Portal and delivered to Amazon's Mechanical Turk workforce marketplace. Only the qualified work will be approved and paid through AMT. The results of the second sub-process (Intelligent Validation and Analysis) can be programmatically returned to CPB Portal.

In the third sub-process Decision Making and Notification, CPB's case administration team uses a decision making template to summarize the investigation results with a recommendation, and attached analysis reports from the previous two sub-processes and deliver them to the decision making team. Once a decision is made, the case administration team will send out notifications to the business cloud consumer who requested this Copyright Protection Service in CPB's business cloud.

It may be a good exercise for the technical professionals to map AMT's core capabilities to the corresponding modules in CCOA. The connection between AMT and Amazon Elastic Compute Cloud (Amazon EC2) [2] can be enabled in CPB's business cloud offering through Amazon web services. However, it is not covered in this paper due to space constraints.

4. Related Work and Discussions

In the area of application or business process provisioning, we have published several papers on the profiling framework [10][16], provisioning and subscription processes, batch requests handling and monitoring [9][17]. In the application development for virtualized Grid computing environment, we have proposed Grid Solution Sphere concept to unleash the business value of Grid Computing [14][15][17]. The practices accumulated from those exercises have been used to articulate the provisioning and application composition aspect of the CCOA.

In the area of service-orientation, SOA Solution Stack (S3) defines an SOA reference architecture in the context of solution design and delivery [8]. The proposed CCOA is also a natural evolution of these practices and S3.

Once a Cloud Computing system is put into operations, there are two major costs associated, i.e., maintenance service fee and energy consumption cost. There is a trend that building new Internet data centers sharply reduces power consumption and the use of floor space. Ensuring Green technologies to be part of the design phase is very important for effective operations for a Cloud Computing platform in the future. IBM Research's 2008 Global Technology Outlook has identified *Internet Scale Data Centers* as an answer to address infrastructure business around Cloud Computing [18].

The purpose of the proposed approach is to bring the power of SOA and virtualization together to help realize and explore business value of Cloud Computing. We would like to encourage the research community to work together to refine CCOA and focus on the theoretical analysis of CCOA to build solid foundations for Cloud Computing. As for future research topics, we will work on the detailed architectural building blocks and interaction patterns for all ten modules defined in CCOA, as well as standard service interfaces (or APIs) between solution-oriented architectural building blocks and their hosting Cloud Computing infrastructure. Through these interfaces, the hosting infrastructure and value-added cloud services or offerings cooperatively determine proper resource provisioning and management actions. For example, a Cloud Computing platform provides basic services (e.g., Hadoop style [5] and beyond) to help build ultra scalable and resilient data processing capabilities in business solutions. Moreover, one can quickly build new or composite services that leverage existing service-oriented assets and applications.

5. Conclusions

In this paper, we have proposed the Cloud Computing Open Architecture (CCOA) based on seven architectural principles and ten architectural modules, by integrating the power of service-oriented architecture (SOA) and virtualization technology of hardware and software. We have also presented an Infrastructure Cloud as a case study to illustrate how to use CCOA to enable infrastructure-level resource sharing as a cloud offering. We have also studied a business cloud to illustrate how to use CCOA to separate the concern of business design from infrastructure enablement through Cloud Publishing Business. The case studies and analysis have shown that the proposed CCOA is an extensible and configurable architecture for providing normative guidance and enabling infrastructure, software, application, and business process sharing in a unified manner.

References

[1] Liang-Jie Zhang, Carl K Chang, Ephraim Feig, Robert Grossman，Keynote Panel, Business Cloud: Bringing The Power of SOA and Cloud Computing，pp.xix, 2008 IEEE International Conference on Services Computing (SCC 2008), July 2008

[2] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/, 2009

[3] XDriver Box service, http://www.box.net/xdrive, 2009

[4] Microsoft Skydrive service, http://skydrive.live.com/, 2009

[5] Hadoop Open Source Project, http://hadoop.apache.org/core/, 2009

[6] OSI Model, http://www.osi.org, 2009

[7] John Y. Sayah, Liang-Jie Zhang, On-demand business collaboration enablement with web services, Decision Support System, 40 (2005), pp.107-127.

[8] Ali Arsanjani, Liang-Jie Zhang, Michael Ellis, Abdul Allam, Kishore Channabasavaiah, "S3: A Service-Oriented Reference Architecture," IT Professional, vol. 9, no. 3, pp. 10-17, May/Jun, 2007

[9] Liang-Jie Zhang, Henry Chang, Tian Chao, Jen-Yao Chung, Zhong Tian, Jingmin Xu, Yingnan Zuo, Shunxiang Yang, Qingyun Ao, A Manageable Web Services Hub Framework and Enabling Technologies for e-Sourcing, IEEE Conference on System, Man, and Cybernetics (SMC'02), 2002.

[10] Shun Xiang Yang, Liang-Jie Zhang, Tian Chao, Jing Min Xu, Ying Nan Zuo, Zhong Tian, and Henry Chang: Adaptive profiling framework and system for service provisioning in e-business solutions. Electronic Commerce Research and Applications 3(2): 139-151 (2004)

[11] Amazon Mechanical Turk, http://aws.amazon.com/mturk/, 2009

[12] Liang-Jie Zhang, Jia Zhang, Hong Cai, Services Computing, Springer and Tsinghua University Press, 2007, ISBN: 978-3-540-38281-2, July 2007

[13] L.-J. Zhang,N. Zhou,Y.-M. Chee,A. Jalaldeen,K. Ponnalagu,R. R. Sindhgatta,A. Arsanjani,and F. Bernardini, SOMA-ME: A platform for the model-driven design of SOA solutions , IBM SYSTEMS JOURNAL, VOL 47, NO 3, 2008, pp.397-413.

[14] Liang-Jie Zhang, Jen-Yao Chung, Qun Zhou, Discover Grid Computing, IBM developerWorks Journal, February 2003, pp.14-19.

[15] Liang-Jie Zhang, Qun Zhou, Jen-Yao Chung, Developing Grid Computing Applications, IBM developerWorks Journal, May 2003, pp.10-15.

[16] Jing Min Xu, Ying Nan Zuo, Shun Xiang Yang, Zhong Tian, Henry Chang, Liang-Jie Zhang, Tian Chao: Membership Portal and Service Provisioning System for an Infrastructure of Hubs: Managed E-Hub. ICEIS (4) 2003: pp. 143-150.

[17] Liang-Jie Zhang, Haifei Li, Herman Lam, Toward a Business Process Grid for Utility Computing, IT Professional ,Volume: 6 , Issue: 5 , Sept.-Oct. 2004, Pages:64 – 63

[18] IBM Research, 2008 Global Technology Outlook, http://www-03.ibm.com/procurement/proweb.nsf/objectdocswebview/file3+-+ibm+gto+overview+-+agerwala/$file/3+-+ibm+gtooverview+-+agerwala.pdf

[19] Google Web Applications for Communication and Collaborations. http://www.google.com/apps

Architecture-Driven Variation Analysis for Designing Cloud Applications

Liang-Jie Zhang [1], Jia Zhang [2]
[1]*IBM T.J. Watson Research Center, USA*
[2]*Department of Computer Science, Northern Illinois University, USA*
[1] *zhanglj@us.ibm.com,*[2] *jiazhang@cs.niu.edu*

Abstract

Service Oriented Architecture (SOA) is one central technical foundation supporting the rapidly emerging Cloud Computing paradigm. To date, however, its application practice is not always successful. One major reason is the lack of a systematic engineering process and tool supported by reusable architectural artifacts. Toward this ultimate goal, this paper proposes a variation-oriented analysis method of performing architectural building blocks (ABB)-based SOA solution design for enabling cloud application design. We present the modeling of solution-level architectural artifacts and their relationships, whose formalization enables event-based variation notification and propagation analysis. We report a prototype tool and describe how we extend the Unified Modeling Language (UML) mechanism to implement the system and enable solution-level variation analysis and enforcement in business cloud as an example.

1. Introduction

Service Oriented Architecture (SOA) is one central technical foundation [1] supporting the rapidly emerging Cloud Computing paradigm, a scalable services delivery and consumption platform in the field of Services Computing [2]. It is a powerful model that allows engineers to dynamically integrate and compose existing service components or services into new cloud services. To date, however, the actual SOA-based application practice is not always a success, because most of SOA practices are conducted at an *ad hoc* manner, mainly based on practitioners' personal experiences. Therefore, our research aims to develop a systematic engineering process and an integrated design tool to facilitate SOA-based variation analysis for cloud application design. Although Cloud Computing comprises various levels of cloud services (infrastructure cloud, software cloud, application cloud, and business cloud), without losing generality, in this research, we focus on SOA at application cloud level. We study how to support and facilitate the design and development of SOA-based applications, that is, SOA solutions.

From industry best practice, layered architectural models have been widely adopted by SOA practitioners to build SOA solutions. For example, IBM has proposed Service-Oriented Solution Stack (S3), a layered enterprise architectural template, as a guidance for IT architects to design the overall architecture of an SOA solution [3]. Nine layers are identified and organized into a two-dimensional model. The horizontal dimension (Operational System layer, Services Component layer, Service layer, Business Process layer, and Service Consumer layer) implements functional requirements; and the vertical dimension (Integration layer, Data Architecture layer, QoS layer, and Governance layer) provides system-support facilities and enablement.

To provide a uniform mechanism for building configurable and reusable SOA solutions on top of S3, we introduced a set of usable Architectural Building Blocks (ABBs) as the fundamental units of an SOA solution [2, 4-6]. An ABB is a component that encapsulates internal states and functions and can be configured and extended. Each layer of S3 is comprised of multiple ABBs, which collaborate to carry expected activities. Considering the adaptive feature of business scenarios, each project may have to configure and customize the layered ABB template library for proprietary requirements and constraints. In addition, the ABB templates imply certain relationships between some ABBs.

This paper reports our research aiming to extend and build the variation analysis [7]-oriented formalism of a modeling environment, with flexibility and extensibility, to enable and support solution-level system design and implementation based on architectural thinking. Specially, we utilize the power of formal methods [8] to model solution-level architectural artifacts and their relationships, whose formalization enables event-based variation notification and propagation analysis. Note that variation here refers to different configuration and customization from architectural artifact template library. Such variation may be propagated and lead to changes in related architectural artifacts.

The solution-level design in the paper refers to metadata-level design of architectural building blocks; and the presented modeling approach is applicable to any layered or tiered architecture. Note the slight differences between several words used throughout the paper, component, architectural artifact, and ABB. A component refers to a comprising part of the structure of an SOA solution with a relatively clearly defined structural boundary. An ABB refers to a highly reusable component in an SOA solution architecture. An architectural artifact

978-0-7695-3840-2/09 $26.00 © 2009 IEEE
DOI 10.1109/CLOUD.2009.86

is a more generic term referring to either a component or an ABB from architecture perspective.

The remainder of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we introduce architectural-centric modeling notations, variation propagation modeling, and solution modeling. In Section 4, we explain our implementation of ABB modeling by extending the UML modeling technique. In Section 5, we introduce our prototype system and explain how it validates a customized SOA solution, as well as an overview of our modeling process. In Section 6, we make conclusions.

2. Related work

Some researchers have studied variation analysis in Web services development [9-11]. Variation is well known as variability management in the generic software development process [12]. Svahnberg et al. define software variability as the ability of a software system to be efficiently extended, changed, customized or configured for use in a particular context [13]. Software variability is usually classified into different categories, from different perspectives such as variation points, variation realization techniques, and feature variations. A variation point refers to a design decision that is consciously left open for software engineers to decide based on some specific situations.

Jiang et al. [9] identify a set of variation points from three resources: WSDL documents, service endpoints, and business logics. However, their modeling stays at simple application examples. The relationships between variation points are not examined.

Kim and Doh [10] identify three tiers (presentation tier, service tier, and application tier) in a Web service design and use UML to model variation points in each tier and their relationships. However, their modeling currently stays at a rudimentary stage and has not been used in any practice.

Ruokonen et al. [11] identify a category of variation needs and types relevant for SOA-based systems, based on two dimensions, system concepts and development process. However, their variation definitions stay at a high level; thus, they do not provide normative guidance for software architects.

Mezini and Ostermann propose a feature-oriented and aspect-oriented modularization approach to manage variability in the context of system families [14].

Various types of formal methods [8] such as Model-Driven Development (MDD) approaches have been widely used to increase software design and development productivity and reduce time-to-market in a systematic manner [15-17]. Representative examples include OMG's MDA [18], Domain Specific Modeling (DSM) [19], and Software Factories [20] from Microsoft.

Mattsson et al. find out that traditional MDD cannot automate the enforcement of architecture on detailed design due to its inability to model architectural design rules [21]. They emphasize the importance of formally modeling architectural design rules. Our reported work in this paper forms a foundation to model architectural artifacts, relationships, and layered verification rules.

Our ABB templates [2, 4-6] provide a comprehensive set of architectural building blocks for each of the nine layers in the S3 model based on industry best practices. The templates can be used as a starting point for software architects to quickly configure and design a prototype for a specific SOA solution. However, although the templates require some relationships between comprising architectural artifacts, these relationships are implicit and require significant learning curves and personal experiences. Therefore, this research aims to build a formalism-based model driven design environment to guide software architects in constructing an appropriate SOA solution.

3. Foundations for ABB modeling

To ease our discussion, Figure 1 shows the recommended ABB configuration for the Service Consumer layer of S3. Eight ABBs are identified: A *consumer* ABB represents an external user. A *presentation (view)* ABB is responsible for communicating to and from external consumers. A *presentation controller* ABB is the coordinator for all other ABBs in the layer. A *consumer profile* ABB is responsible for getting customer-specific information. An *access control* ABB provides authentication/authorization capabilities. A *format transformation* ABB is responsible for translation of query content format. A *configuration rule* ABB is responsible for hosting rules that define how the ABBs can be configured. A *cache* ABB is responsible for temporarily storing consumer interaction-related data. Detailed ABB definition and configuration can refer to our previous report [6].

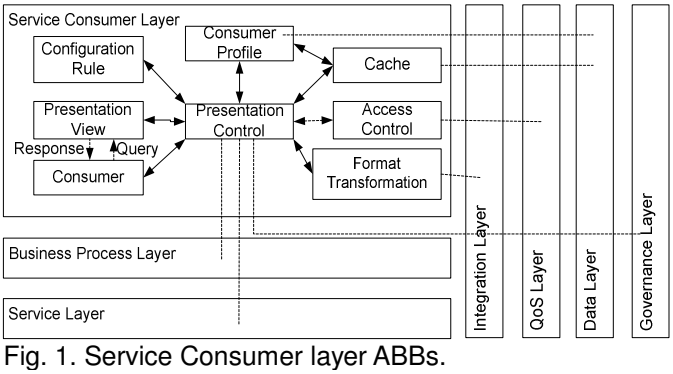


Fig. 1. Service Consumer layer ABBs.

3.1 ABB modeling

To facilitate our formalization, Table 1 summarizes the fundamental notations and their semantic meanings, which will be discussed in detail in the following sections and used throughout the paper.

TABLE 1
NOTATIONS OF ABB MODELING

Notations	Semantic Meanings
X	a layer in S3
Φ_X	ABB type set in layer X
Ω_X	ABB instance set in layer X
A_{ABB}^X	ABB type A at layer X
$a^{ABB,A}$	an instance of ABB type A (i.e., with stereotype A)
$*_A^X$	# of instances of A_{ABB}^X
$\mathrm{M}_m^n\left\langle A_{ABB}^X, B_{ABB}^Y \right\rangle$	ABB type A and ABB type B has an m:n mapping relationship
$\mathrm{M}_p^q\left\langle a^{ABB,A}, b^{ABB,B} \right\rangle$	an instance of ABB type A and an instance of ABB type B has a p:q mapping relationship
M_p^q	a p:q mapping relationship between either a pair of ABB types or a pair of ABB instances
$\subseteq$	compliant relationship between ABB instances with that between corresponding ABB types (both mapping and propagation)
$\mathrm{P}_{\mathrm{M}_p^q}^{R_{\mapsto}^t}\left(a^{ABB,A}, b^{ABB,B}\right)$	protocol between a pair of ABB instances (M_p^q for mapping and $R_{\mapsto}^t$ for propagation)
$\mathrm{P}_{\mathrm{M}_m^n}^{R_{UML}}\left(A_{ABB}^X, B_{ABB}^Y\right)$	protocol between a pair of ABB types (M_m^n for mapping and R_{UML} relationship in UML 2.0)
S	an SOA solution
$a^{ABB,A} \mapsto b^{ABB,B}$	a variation publisher/subscriber relationship exists from $a^{ABB,A}$ to $b^{ABB,B}$
$R_{\mapsto}^t$	type *t* publisher/subscriber relationship
$\prec$	Compliant protocol between a pair of ABB instances with their corresponding ABB types

Definition 1. An *Architectural Building Block (ABB)* is a component in S3 that encapsulates internal states and functions and can be configured and extended. An *ABB type* refers to an ABB we identified representing a typical fine-grained building class in an SOA solution. An *ABB instance* refers to a building block in a specific SOA solution implementing an ABB type. An *ABB component* refers to either an ABB type or an ABB instance.

Definition 2. A *mapping relationship* refers to the cardinality between two ABB components, if there is relationship between them. A mapping relationship can be represented by a pair of sets m:n, each falling into one of the three possibilities: {1}, {0..*}, or {1..*}.

$$\mathrm{M}_m^n \Rightarrow \mathrm{M}_m^n \in \left\{null, \mathrm{M}_1^1, \mathrm{M}_1^{0..*}, \mathrm{M}_1^{1..*}, \mathrm{M}_{0..*}^1, ...\right\} \quad \text{(R1)}$$

The cardinality definitions can be used to detect and verify component-level variations. For the same reason, we define a *null* relationship to imply that no relationship exists between two ABB components.

Definition 3. A *compliant mapping relationship* requires that the mapping relationship between a pair of ABB instances be either the same as or stronger than the one defined between their corresponding ABB types:

$$\mathrm{M}_p^q\left\langle a^{ABB,A}, b^{ABB,B} \right\rangle \subseteq \mathrm{M}_m^n\left\langle A_{ABB}^X, B_{ABB}^Y \right\rangle \Rightarrow$$
$$\left((q \subseteq n) \wedge (p \subseteq m)\right) \vee \left(\mathrm{M}_p^q = null \wedge \mathrm{M}_m^n = null\right) \quad \text{(R2)}$$

3.2 Variation propagation modeling

3.2.1. Propagation relationship

To maintain loose coupling between ABBs in an SOA solution, we applied the publisher/subscriber design pattern [22] to manage variation propagation and synchronize cooperative ABBs. By notifying all of its subscribers about changes, a publisher enables single-directional change propagation.

Definition 4. For each identified single-directional relationship between a pair of ABB instances, the ABB instance that is independent from the other one is assigned as a variation publisher; the ABB instance that depends on the other one is assigned as a variation subscriber. Their relationship can be represented by: $a^{ABB,A} \mapsto b^{ABB,B}$, where: $a^{ABB,A}$ and $b^{ABB,B}$ denote two ABB instances, with the former denoting the variation publisher in the relationship and the latter denoting the variation subscriber.

The propagation relationships between each pair of ABBs can be normalized as one or two single-directional relationships (a bi-lateral relationship can be broken into two single-lateral relationships). Such a relationship is enabled by the variation subscriber registering itself at the variation publisher.

3.2.2. Propagation between ABB types

Properly designed propagation relationships between ABB types can be used for two purposes. First, since metamodel is typically maintained by the highest-level architects, the defined relationships can be used as guidance for actual model design. Second, defined relationships can be used to validate and verify actual propagation relationships between ABB instances.

Propagation relationships can be overwritten at ABB instances from ABB types within certain degrees if necessary. For example, a two-dimensional association relationship can be strengthened into one-dimensional directed association. We summarize our considerations of propagation relationships between ABB types and ABB

instances in Figure 2.

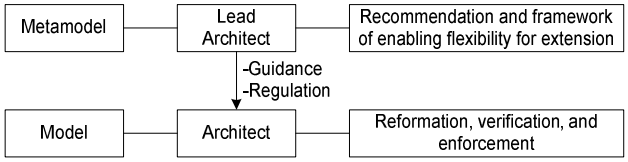


Fig. 2. Propagation relationship specification model.

As shown in Figure 2, metamodels are typically maintained by lead architects and models are constructed by individual SOA architects. Our strategy is that metamodels can leverage and extend normal UML 2.0 relationships. The categorized propagation relationships summarized in Table 2 can be recommended but do not have to be enforced. On the other hand, models should be allowed to use the propagation relationships in Table 2. While architects could still try to use UML 2.0 relationships, some reformation work will be performed underlying (e.g., break a two-dimensional association into two one-dimensional associations), and incompatible relationships (e.g., package import) will be warned and enforced to be removed from a valid model.

TABLE 2.
ABB-LEVEL VARIATION MANAGEMENT.

Cat	Covered UML relationships	Descriptions	Variation management
1	directed association, dependency, usage	Customized dependency relationship	Registration at the depended ABB publisher
2	aggregation, composition	Whole/part relationship	Registration at the whole ABB (publisher)
3	implements, realization	Abstraction/implementation relationship	Registration at the abstraction ABB publisher)
4	generalization	Super-/sub-class relationship	Registration at the sub-class ABB publisher

Definition 5. A compliant propagation relationship refers to the propagation relationship defined between a pair of ABB instances that must be compliant with that defined between the corresponding pair of ABB types. The requirement can be represented as R3:

$$\forall\left(R^t_{\mapsto}\left\langle a^{ABB,A}, b^{ABB,B}\right\rangle\right)\Rightarrow$$
$$R^t_{\mapsto}\left\langle a^{ABB,A}, b^{ABB,B}\right\rangle \subseteq R_{UML}\left\langle A^X_{ABB}, B^Y_{ABB}\right\rangle \quad \text{(R3)}$$

As mentioned earlier in this paper, we considered simplified publisher/subscriber relationship. Thus, R3 can be simplified into R4:

$$\forall\left(R_{\mapsto}\left\langle a^{ABB,A}, b^{ABB,B}\right\rangle\right)\Rightarrow$$
$$R_{\mapsto}\left\langle a^{ABB,A}, b^{ABB,B}\right\rangle \subseteq R_{UML}\left\langle A^X_{ABB}, B^Y_{ABB}\right\rangle \quad \text{(R4)}$$

3.3 SOA solution definition

In this section, we discuss how to model an SOA solution based on ABB propagation modeling.

3.3.1. Propagation path

When an ABB-related variation occurs at runtime, it needs to be propagated to all related ABB instances.

Definition 6. A propagation set depicts an ordered set of all either directly or indirectly impacted ABB instances in an SOA solution when a variation happens at one contained ABB instance. A propagation set can be denoted by a directed graph represented by a 4-tuple $G^P_a = (PN, PA, a, PSet)$, where PN represents a set of ABB instances; PA represents a set of directed arcs; a represents the starting point of ABB instance; and $PSet$ represents a set that includes all propagation paths starting from a. Each arc links two nodes if there is a propagation relationship between the two ABB instances. A propagation path of the ABB instance a depicts a subset of its propagation set, starting from a and cannot further propagate. A propagation path of an ABB instance a can be denoted by a directed graph $path = (\vec{N}, \vec{A}, a)$,

where: $\vec{N} = \{a, N_1, N_2, ..., Nn\}$,

$\vec{A} = \left\{R^t_{\mapsto}(a, N_1), R^t_{\mapsto}(N_1, N_2), R^t_{\mapsto}(N_2, N_3), ...R^t_{\mapsto}(N_{n-1}, N_n)\right\}$,

$\vec{N} \subseteq PN, \vec{A} \subseteq PA$

Definition 5 implies that:

$\forall \alpha \in PA \Rightarrow$

$$\exists\left(\left(a^{ABB,A} \in N\right) \wedge \left(b^{ABB,B} \in N\right) \wedge \left(R^t_{\mapsto}\left\langle a^{ABB,A}, b^{ABB,B}\right\rangle\right)\right) \quad \text{(R5)}$$

and

$PSet = \{P_1, P_2, ..., P_k\}, P_i = \{a, N_1, N_2, ..., N_n\},$

$i \in [1, k], N_j \in PN, j \in [1, n], a \mapsto N_1, N_1 \mapsto N_2, ..., N_{n-1} \mapsto N_n$

A propagation path depicts a possible farthest propagation pathway starting from an ABB instance. A propagation points out a message passing path between involved ABB instances at runtime.

3.3.2. ABB protocol

Putting together mapping relationship and propagation relationship, we can define protocols between ABB types and ABB instances.

Definition 8. One protocol between a pair of ABB types contains two elements: a mapping relationship (as defined in Definition 1) and a propagation relationship referring to a UML 2.0 relationship that defines the correlation between two ABB types. Such a protocol can be represented as follows:

$\mathrm{P}_{\mathrm{M}_m^n}^{R_{UML}}\left(A_{ABB}^X, B_{ABB}^Y\right)$, where:

A_{ABB}^X and B_{ABB}^Y denote two ABB types; M_m^n denotes that the cardinality between the two ABB types is $M_m^n\left(A_{ABB}^X, B_{ABB}^Y\right)$; there is a correlation relationship between the two ABB types that can be represented by a UML 2.0-defined relationship R_{UML}.

Definition 9. One protocol between a pair of ABB instances contains two elements: a mapping relationship (as defined in Definition 1) and a propagation relationship referring to a publisher/subscriber relationship that defines the direction of variation propagation. Such a protocol can be represented as follows:

$\mathrm{P}_{\mathrm{M}_p^q}^{R_{\mapsto}^t}\left(a^{ABB,A}, b^{ABB,B}\right)$, where:

$a^{ABB,A}$ and $b^{ABB,B}$ denote two ABB instances; M_p^q denotes that the cardinality between the two ABB instances is $M_p^q\left(a^{ABB,A}, b^{ABB,B}\right)$; there is a publisher/subscriber relationship between the two instances $a^{ABB,A} \mapsto b^{ABB,B}$, where $a^{ABB,A}$ is the publisher and $b^{ABB,B}$ is the subscriber; and the publisher/subscriber type $R_{\mapsto}^t$ between the two ABB instances is of type t $R_{\mapsto}^t$ (In this paper since we only consider one type of generic publisher/subscriber relationship, it can be simplified as $R_{\mapsto}$).

Definition 10. A compliant ABB protocol refers to the protocol defined between a pair of ABB instances must be compliant with that defined between the corresponding pair of ABB types in two aspects: one is their mapping relationship must be compliant; the other one is their propagation relationship must be compliant.

$$\mathrm{P}_{\mathrm{M}_m^n}^{R_{UML}}\left(A_{ABB}^X, B_{ABB}^Y\right) \prec \mathrm{P}_{\mathrm{M}_p^q}^{R_{\mapsto}^t}\left(a^{ABB,A}, b^{ABB,B}\right) \Rightarrow \left(R_{\mapsto}^t \subseteq R_{UML}\right) \wedge \left(\mathrm{M}_p^q \subseteq \mathrm{M}_m^n\right) \quad \text{(R6)}$$

3.3.3. Definition of an SOA Solution

Definition 11. An SOA solution S is defined as a tuple as follows: $S = \langle \Phi, \Omega \rangle$, where:

- Φ represents the metamodel of the solution, which is in turn a tuple: $\Phi = \langle \Phi_\Phi, \Phi_P \rangle$, where: Φ_Φ represents nine layers of ABB types: $\Phi_\Phi = \bigcup_{i=1}^{9} \Phi_{X_i}$, Φ_P represents the relationships between the defined ABB types:
- Ω represents the actual model of the solution, which is in turn a tuple: $\Omega = \langle \Omega_\Omega, \Omega_P \rangle$, where: Ω_Ω represents nine layers of ABB instances: $\Omega_\Omega = \bigcup_{i=1}^{9} \Omega_{X_i}$, Ω_P represents the relationships between the defined ABB instances.

4. SOA solution modeling

Based on the presented formal ABB modeling, we now discuss how we leveraged the model-driven approach to extend UML to implement ABB-based SOA solution modeling as an example. UML has become a widely accepted industry standard for software system modeling [23]. The major technical challenge we encountered is how to implement our formal ABB representation using UML modeling. Particularly, we had to tackle two issues: how to model variation propagation between ABB instances and how to model ABB-based SOA solutions.

4.1. Propagation between ABB instances

UML provides a comprehensive set of relationship types to model relationships between entities (e.g., class, interface, component, and package) [9]. In this paper, we are only interested in the relationship types that may lead to variation propagation through ABB instances. For example, consider a *Presentation Controller* ABB that *uses* an *Access Control* ABB. If an access control method name in the *Access Control* ABB is changed, the *Presentation Controller* ABB should be informed to make consequent changes.

We examined all relationship types defined in UML 2.0. Some relationship types will not cause component-level variation propagation (e.g., substitution), or have equivalents (e.g., abstraction), or are irrelevant to the semantics of ABB instances (e.g., instantiation and binding). At ABB level, these relationships will not be allowed so we remove them from our consideration. After this process, we obtain a set of 10 relationships: association, link, directed association, aggregation, composition, generalization, implements, realization, dependency, and usage.

Among the remaining set, some relationship types are one-directional (e.g., directed association and aggregation) while others are bi-directional (association and link). Because we intend to analyze variation propagation, direction is important here. Therefore, for each bi-directional relationship, we break them into two single-directional ones.

Meanwhile, considering component-level variation propagation, some relationships will cause the same or similar action. We organize the studied relationships into four categories as shown in Table 2. Detailed definition

for each UML relationship can be found from UML 2.0 specification [23] and will be followed in our study.

The major reason why we categorize the relationships between ABBs is for enabling variation propagation analysis and management, which is also the criterion of how we categorize the relationships. Category 1 represents customized dependency relationship between two ABBs, which covers three types of UML relationships: *directed association*, *dependency*, and *usage*. Category 2 represents whole/part relationship between two ABBs, which covers two types of UML relationships: *aggregation* and *composition*. Category 3 represents abstraction/implementation relationship between two ABBs, which covers two types of UML relationships: *implements* and *realization*. Category 4 represents super-/sub-class relationship between two ABBs, which covers the UML relationship *generalization*.

Each category implies a proprietary variation management approach. Table 2 provides the guidance on how to apply the publisher/subscriber pattern and assign variation publisher/subscriber roles to each type of ABB relationship. The strategy is that the ABB that will be notified for variation propagation will be assigned as a variation subscriber; and the other ABB initiating a variation will be assigned as a variation publisher. For Category 1, each covered UML relationship implies a dependency relationship between the two ABBs. Therefore, the depended ABB is assigned as the variation publisher and the dependent ABB is assigned as the variation subscriber; the dependent ABB registers itself at the depended ABB. For Category 2, the ABB as a part is assigned as the variation subscriber and the ABB as the whole is assigned as the variation publisher; the part ABB registers itself at the whole ABB. For Category 3, the ABB as the abstraction is assigned as the variation publisher and the ABB as the implementation is assigned as the variation subscriber; the implementation ABB registers itself at the abstraction ABB. For Category 4, the ABB as the sub-class is assigned as the variation publisher and the ABB as the super-class is assigned as the variation subscriber; the super-class ABB registers itself at the sub-class ABB.

Six levels of variations should be propagated: data entity, message, business primitive, business construct, business protocol, and business process [2]. All of these variations can be propagated through the same publisher/subscriber approach. However, it is useful to differentiate between levels of variations through propagation, so that proper action can be conducted effectively and efficiently. For example, if a variation happened at a variation publisher is at an operation level, its variation subscriber may only update its corresponding operation invocation. If the propagation only informs a variation occurrence without variation level, though, the variation subscriber may not be able to take right action promptly. This simple example illustrates the necessity of differentiating variation levels. To address this need, our solution is to extend the original publisher/subscriber pattern by associating variation level information. When a publisher/subscriber relationship is set up between two ABB instances, one of the six levels of variation will be bound to the publisher/subscriber relationship. To simplify the studied domain, without losing generality, in this paper we only consider the original default publisher/subscriber format. In other words, when any level of variation happens at a variation publisher, its corresponding variation subscribers will be informed in the same way. Studying variation propagation at various levels in detail will be our future research.

4.2. ABB-based SOA solution modeling

In this section, we introduce how we extend UML to model ABBs associated with variations in an SOA solution. Starting from UML 2.0, metamodels specify the abstract syntax and semantics of UML modeling concepts. On top of metamodel, profiles and constraints provide mechanisms to extend the existing UML metamodel to model specific applications. In more detail, the UML metamodel defines the basic stereotypes (i.e., data types) that users can utilize to build UML models. Examples of basic stereotypes are class, package, and association. A user-defined metamodel defines newly added stereotypes and their relationships. A profile carries stereotypes and constraints can be applied by users to create instances of stereotypes in UML models. In other words, a user-defined profile defines application-specific building blocks that users can use to build specific UML models.

In this research, we extend the UML metamodel to model ABB-based SOA solutions. Each ABB is modeled as a UML stereotype; stereotypes representing all ABBs defined in one layer are grouped in one package as a profile. The rationale is that, every building block in an SOA solution is an instance of an identified ABB. In other words, an ABB can be considered as a class with specific definitions that can be instantiated into final building blocks.

As S3 illustrates, a typical SOA solution can be organized into a nine-layer structure. The final code structure can consequently be organized into nine packages. Note that these nine packages only represent the top-level software structure; more packages can be identified if needed. Therefore, we establish a mapping between SOA solutions with UML models, by mapping an S3 layer to a UML package and mapping an ABB to a UML stereotype.

4.3. Metamodel-centered adaptive modeling

According to our extensions to UML to model SOA solutions using ABBs, we created three kinds of artifacts:

metamodel, profile, and palette. For each layer of the S3 model, we first created a metamodel representing all identified ABBs as stereotypes as the conceptual model of the layer, then we created a package of profiles representing each ABB as a stereotype, and then we created a palette containing all ABBs as building blocks for users to use to create UML models.

We found that there are inherent relationships among the three concepts: metamodel, profile, and palette. As a matter of fact, all of the three concepts define stereotypes that can be used by users to build UML models. A palette is a typical way for a UML modeling tool to prepare a set of available stereotypes for users to pick from and draw UML diagrams. A profile carries defined stereotypes for users to apply to models. A metamodel is the source to formally define stereotypes.

Due to the intrinsic relationships among the three concepts, if we create them separately, whenever we make changes to any ABBs, we have to apply the same changes to all three types. This approach of manual consistency control is not only inefficient but also error-prone.

Our solution is to utilize the inherent relationships among the three concepts and use the metamodel as the basis to realize automatic synchronization among them. The details of the idea can be summarized as in the pseudo code shown below. We first construct the metamodel for each layer. Then for each metamodel, we identify the stereotypes and create a corresponding profile containing all ABBs as stereotypes. Afterwards we create a corresponding palette comprising all ABBs as building blocks. Any UML modeling environment typically already contains a built-in palette representing basic building blocks in hierarchies for UML modeling, such as class, association, class diagram, and so on. To differentiate between our SOA solution-oriented palette and embedded-in palette, we organize palette in model libraries instead.

Alg. 1: Metamodel-centered SOA solution model management

1. If they do not exist, create all three categories of elements.
   1.1 For each of the nine layers of the S3 model,
   Create a metamodel with all identified ABBs;
   1.2. For each of the nine metamodel,
   1.2.1. Create a profile;
   1.2.2. Iterate through all ABBs in the metamodel,
   Create a stereotype in the profile;
   1.3. For each of the nine metamodel,
   1.3.1. Create a model library;
   1.3.2. Iterate through all ABBs in the metamodel,
   Create a building block in the model library;
2. If already existed but a change is required,
   2.1 make changes to the corresponding metamodel;
   2.2 re-generate the corresponding profile elements only;
   2.3 re-generate corresponding model library elements.

Whenever changes are required, we will modify the corresponding metamodels. Afterwards, by re-generating corresponding profile elements and model library elements, all three types of model elements can be synchronized. It should be noted that intelligence is necessary here to ensure that only involved profile elements and model library elements be re-generated, so that other elements with customized code (e.g., with user-added derived stereotypes) can stay untouched.

4.4. Usages of metamodel for generating solution patterns

The industry best practice we use not only leads to the identification of ABBs, but also depicts the relationships between the ABBs. As shown in Figure 3, since a metamodel is a class diagram, these relationships can be represented in the metamodel using standard UML relationship elements. For example, there is a one-to-many relationship between a "Presentation Controller" stereotype and a "Presentation" stereotype. These relationships can help to build SOA solutions using instances of ABBs (i.e., stereotypes). Figure 3 shows an example structure of a Services Consumer layer of an SOA solution. As shown in Figure 3, each of the ABB identified has an instance; these ABB instances are interconnected according to predefined relationships.

5. SOA solution validation

In this section, we discuss how to validate a customized SOA model. We organize architectural rules in a three-level hierarchy: (1) system-level rules, (2) layer-level rules, and (3) ABB-level rules. A set of top-level rules is established at global S3 model-wide for any SOA solution. These rules are to be enforced with the highest priority over any other rules. For example, each ABB type can only belong to one layer. Each layer can define a specific set of rules as layer-level rules, which have to be enforced with the second highest priority. ABB-level rules refer to the relationships defined between ABB types that have to be enforced on corresponding ABB instances.

5.1. System-level rules

Definition 12. The system-level rule set for an SOA solution $S = \langle \Phi, \Omega \rangle$ contains six predefined rules:

1. $\forall A_{ABB}^{X} \Rightarrow \exists(\Phi_X \in \Phi_\Phi) \wedge \left(A_{ABB}^{X} \in \Phi_X\right) \wedge \left(\exists A_{ABB}^{X} \wedge \exists A_{ABB}^{Y} \Rightarrow X = Y\right)$
2. $\forall a^{ABB,A} \Rightarrow \exists(\Omega_X \in \Omega_\Omega) \wedge \left(a^{ABB,A} \in \Omega_X\right) \wedge \left(\exists a^{ABB,A} \in \Omega_X \wedge a^{ABB,A} \in \Omega_Y \Rightarrow X = Y\right)$
3. $\Phi_X = null \Rightarrow \Phi_X = \phi$

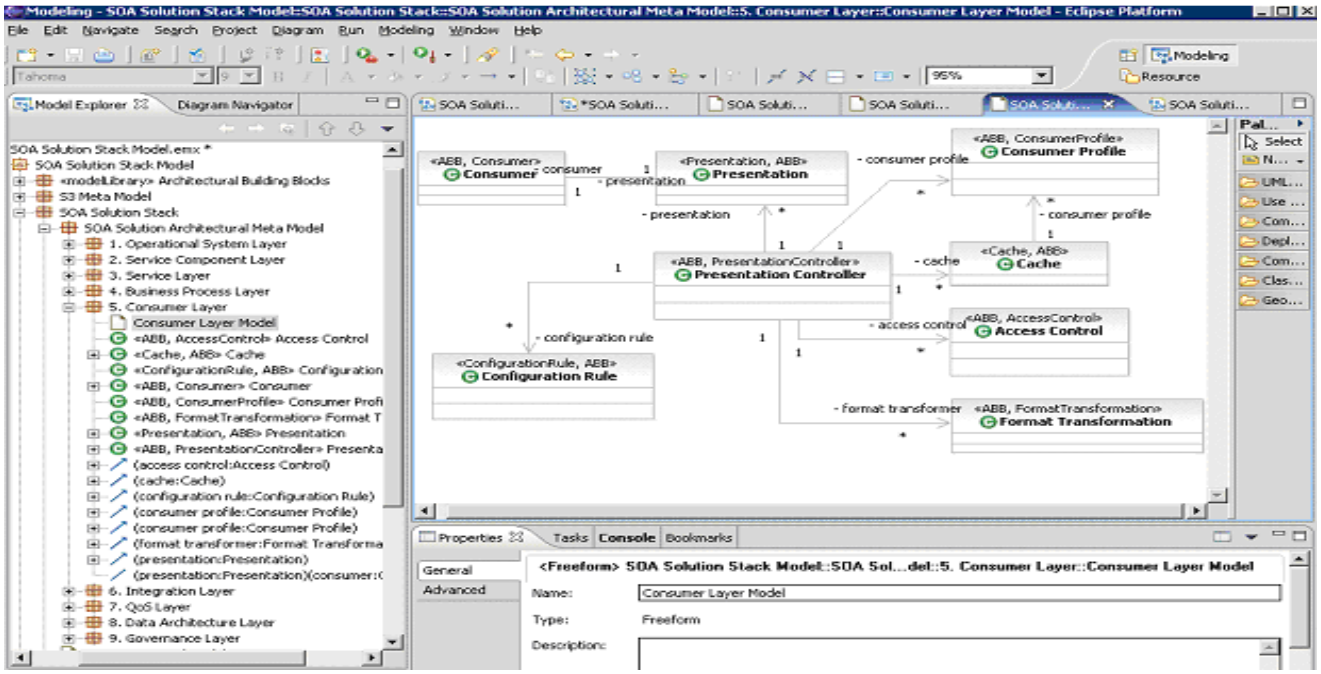


Fig. 3. Generated default modeling template.

4. $\Phi_X = \phi \Rightarrow \Omega_X = \phi$

5. $*_A^X \geq 0$

$$\forall\left(\mathrm{P}_{\mathrm{M}_m^n}^{R_{UML}}\left\langle A_{ABB}^X, B_{ABB}^Y\right\rangle \wedge \mathrm{P}_{\mathrm{M}_p^q}^{R_{\mapsto}^t}\left\langle a^{ABB,A}, b^{ABB,B}\right\rangle\right) \Rightarrow$$

6.

$$\mathrm{P}_{\mathrm{M}_m^n}^{R_{UML}}\left(A_{ABB}^X, B_{ABB}^Y\right) \prec \mathrm{P}_{\mathrm{M}_p^q}^{R_{\mapsto}^t}\left(a^{ABB,A}, b^{ABB,B}\right)$$

Rule 1 denotes that each ABB type A_{ABB}^X belongs to and only belongs to one specific layer X. Rule 2 denotes that each instance of ABB type A belongs to and only belongs to one layer X to which the ABB type A belongs. Rule 3 denotes that for a particular SOA solution, one layer may contain no ABBs, meaning that the layer is not necessary for the solution. Rule 4 denotes that if a specific layer does not exist in the metamodel, the layer does not exist in the actual SOA model. Rule 5 denotes that one ABB type A may or may not have instances in a specific SOA solution, and one ABB type A may have more than one instance in a specific SOA solution. Rule 6 denotes that a protocol between a pair of ABB instances in a specific SOA solution is required to be compliant with that defined between the corresponding ABB types in the corresponding metamodel of the SOA solution. In general, the top-level rules are implicitly defined and cannot be modified throughout the modeling of any SOA solution.

5.2. Layer-level rules

Each layer in the S3 model can carry a specific set of rules as layer-level rules. Taking the Service Consumer layer as an example, its layer-level rule set possesses one rule R6:

$$\Omega_{Consumer} \neq \phi \Rightarrow$$

$$iff\left(\begin{array}{l} presentation\_controller^{ABB,\mathrm{Pr}esentation\_Controller} \\ \in \Omega_{Consumer\_Layer} \wedge *_{presentation\_controller}^{Consumer\_Layer} = 1 \end{array}\right) \quad (R6)$$

R6 denotes that if the Service Consumer layer model is not empty, then there has to be one and only one instance of *Presentation Controller* ABB exists. This rule can be enforced by setting the multiplicity of the *Presentation Controller* ABB type to be "1."

Each layer may carry some predefined layer-level rules. Typically, these layer-level rules can be modified by chief architects at the time of construction of a metamodel for a specific SOA solution. For example, a comprehensive application may allow multiple instances of *Presentation Controller* ABB exist in its Service Consumer layer. This rule can be represented as R7 and can be set up by setting the multiplicity of the *Presentation Controller* ABB type to be "1..*".

$$\Omega_{Consumer} \neq \phi \Rightarrow$$

$$iff\left(\begin{array}{l} presentation\_controller^{ABB,\mathrm{Pr}esentation\_Controller} \\ \in \Omega_{Consumer\_Layer} \wedge *_{presentation\_controller}^{Consumer\_Layer} \geq 1 \end{array}\right) (R7)$$

Meanwhile, application-specific layer-level rules can be added if necessary. For example, one particular PDA-oriented application may require that at least one pair of

(PDA_Consumer, PDA_Presentation) instances are created for the ABB type pair (Consumer, Presentation), meaning that the solution has to support PDA users by providing corresponding PDA-specific interface presentation generation mechanisms. The rule can be represented as follows:

$$\Omega_{\mathrm{Pr}esentation\_Layer} \wedge *^{\mathrm{Pr}esentation\_Layer}_{Consumer} \geq 1 \qquad \text{(R8)}$$

5.3. ABB-level rules

ABB-level rules define the mutual connections between ABB types that need to be enforced on corresponding ABB instances. The connection relationship in our concern includes cardinality relationship and propagation relationship. Such relationships can be deduced and transformed into rules to regulate the construction of SOA models using ABB instances. For example, a chief architect may define that there is a 0:m dependency relationship between a *Presentation Controller* ABB type and an *Access Controller* ABB type in the Service Consumer layer, which can be represented as follows:

$$\mathrm{P}^{R_{Dependency}}_{\mathrm{M}_1^{0..*}} \left(\begin{array}{l} \mathrm{Pr}esentation\_Controller^{Consumer\_Layer}_{ABB}, \\ Access\_Controller^{Consumer\_Layer}_{ABB} \end{array} \right) \text{(R9)}$$

The above rule can be deduced into the following set of rules comprising three rules regarding related ABB instances:

$$(1)\ \mathrm{P}^{R_{\mapsto}}_{\mathrm{M}_1^{0..*}} \left(\begin{array}{l} presentation\_controller^{ABB,Consumer\_Layer}, \\ access\_controller^{ABB,Consumer\_Layer} \end{array} \right)$$

$$(2)\ \mathrm{P}^{R_{\mapsto}}_{\mathrm{M}_1^{1..*}} \left(\begin{array}{l} presentation\_controller^{ABB,Consumer\_Layer}, \\ access\_controller^{ABB,Consumer\_Layer} \end{array} \right)$$

$$(3)\ \mathrm{P}^{R_{\mapsto}}_{\mathrm{M}_1^{1}} \left(\begin{array}{l} presentation\_controller^{ABB,Consumer\_Layer}, \\ access\_controller^{ABB,Consumer\_Layer} \end{array} \right)$$

The deducted rules indicate acceptable relationships regarding corresponding ABB instances in an SOA model. In other words, if an SOA model contains a pair of ABB instances *Presentation Controller* and *Access Controller*, their relationship has to be one of the above three possibilities to be considered as a valid relationship.

Using the same method, each defined relationship between a pair of contained ABB types in a metamodel can be represented as an ABB-level rule; each such rule can be deducted into a set of rules for a corresponding pair of ABB instances. Accumulating them all, for each metamodel for an S3 layer, we can obtain a knowledge base containing a set of relationship rules for acceptable relationships for the S3 layer.

In summary, each relationship defined between a pair of ABB types in a metamodel can be deducted into a set of rules for ABB instances. Each defined relationships between a pair of ABB instances can be transformed into a normalized rule to be compared with the deducted rule sets. If it can be found in the knowledge base, it is a valid relationship; otherwise, it should be rejected.

5.4. Modeling environment and process

With the establishment of ABB-based modeling technique and solution validation mechanism, we constructed SOA Modeling Environment (SOA-ME) as a prototype modeling tool. It is built upon IBM Rational Software Architect (RSA) as its development platform to exploit the comprehensive architectural design artifacts and interfaces of the RSA.

SOA-ME can be logically divided into three sections: rule generation, static validation, and dynamic validation. The rule generation section creates system-level rules, layer-level rules, and ABB-level rules into the *rule knowledge base*. The static validation section takes the input of a customized SOA model, analyzes its contents, while checking against the rule knowledge base. The result of the static model parser is a valid SOA model, which can be deployed for runtime usages. The dynamic validation section examines the messages passed between ABB instances and re-engineers the propagation paths. Then it compares the generated sets with the stored propagation paths created at the static validation phase. If the two sets are equivalent, the validation result is positive.

6. Conclusions

In this paper, we present our design and development of a systematic SOA solution modeling and variation propagation analysis and ABB template library. The architectural artifacts and variation propagation modeling method enables software architects to manage SOA solution-level quality. Our research lays a foundation to build a practical engineering tool to guide software architects in designing SOA solutions toward application cloud services. The formalization of variation-oriented analysis for architectural artifacts allows solution-level enforcement and performance analysis, and supports architectural evolutionary changes.

For our future study, we plan to explore rule-based formal variation propagation analysis and verification algorithms in the context of our modeling tool. In addition to compile-time static verification, we plan to study dynamic variation analysis to provide solution adaptability for run-time evolutionary changes in service provisioning scenarios for Cloud Computing. It is noted that the presented modeling approach can be applied to any layered or tiered architecture.

7. References

[1] L.-J. Zhang and Q. Zhou, "CCOA: Cloud Computing Open Architecture", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, 2009, Los Angeles, CA, USA, pp.

[2] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer, 2007.

[3] A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah, "S3: A Service-Oriented Reference Architecture", *IT Professional*, May, 2007: pp. 10-17.

[4] L.-J. Zhang and J. Zhang, "Design of Service Component Layer in SOA Reference Architecture", in Proceedings of *33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, Jul. 20-24, 2009, Seattle, WA, USA, pp. 474-497.

[5] L.-J. Zhang and J. Zhang, "Componentization of Business Process Layer in The SOA Reference Architecture", in Proceedings of *IEEE International Conference on Services Computing (SCC)*, 2009, Bangalore, India.

[6] L.-J. Zhang, J. Zhang, and A. Allam, "A Method and Case Study of Designing Presentation Module in an SOA-based Solution Using Configurable Architectural Building Blocks (ABBs)", in Proceedings of *IEEE International Conference on Services Computing (SCC)*, Jul. 8-11, 2008, Honolulu, HI, USA, pp. 459-467.

[7] L.-J. Zhang, A. Arsanjani, A. Allam, D. Lu, and Y.-M. Chee, "Variation-Oriented Analysis for SOA Solution Design", in Proceedings of *IEEE International Conference on Services Computing (SCC)*, Jul. 9-13, 2007, Salt Lake City, UT, USA, pp. 560-568.

[8] I. Traore and D.B. Aredo, "Enhancing Structured Review with Model-Based Verification", *IEEE Transactions on Software Engineering*, 2004, 30(11): pp. 736-753.

[9] J. Jiang, A. Ruokonen, and T. Systa, "Pattern-Based Variability Management in Web Service Development", in Proceedings of *Third IEEE European Conference on Web Services (ECOWS)*, Nov. 14-16, 2005, Växj, Sweden, pp. 83-94.

[10] Y. Kim and K.-G. Doh, "Adaptable Web Services Modeling using Variability Analysis", in Proceedings of *Third 2008 International Conference on Convergence and Hybrid Information Technology (ICCIT)*, Nov. 11-13, 2008, Busan, South Korea, pp. 700-705.

[11] A. Ruokonen, V. Räisänen, M. Siikarla, K. Koskimies, and T. Systä, "Variation Needs in Service-Based Systems", in Proceedings of *2008 Sixth European Conference on Web Services (ECOWS)*, Nov. 12-14, 2008, Dublin, Ireland, pp. 115-124.

[12] M. Jazayeri, A. Ran, and F.v.d. Linden, *Software Architecture for Product Families: Principles and Practice.* Addison-Wesley, 2000.

[13] M. Svahnberg, J.v. Gurp, and J. Bosch, "A Taxonomy of Variability Realization Techniques", *Software Practice and Experience*, 2005, 35(8): pp. 705-754.

[14] M. Mezini and K. Ostermann, "Variability Management with Feature-Oriented Programming and Aspects", in Proceedings of *the 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Oct. 31-Nov. 6, 2004, Newport Beach, CA, USA, pp. 127-136.

[15] M.J. Escalona and G. Aragón, "NDT. A Model-Driven Approach for Web Requirements", *IEEE Transactions on Software Engineering*, May, 2008: pp. 377-390.

[16] D.C. Schmidt, "Model-Driven Engineering", *IEEE Computer*, Feb., 2006: pp. 25-31.

[17] S. Sendall and W. Kozaczynski, "Model Transformation: The Heart and Soul of Model-Driven Software Development", *IEEE Software*, Sep., 2003: pp. 42-45.

[18] OMG, "Mda Guide Version 1.0.1", Jan. 6, 2003, Available.

[19] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, "Model-Integrated Development of Embedded Software", *Proceedings of the IEEE*, 2003, 91(1): pp. 145-164.

[20] J. Greenfield and K. Short, *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley Pub., Indianapolis, IN, USA, 2004.

[21] A. Mattsson, B. Lundell, B. Lings, and B. Fitzgerald, "Linking Model Driven Development and Software Architecture: A Case Study", *IEEE Transactions on Software Engineering*, Oct., 2008: pp. 1-12.

[22] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System Of Patterns*. John Wiley & Sons Ltd., West Sussex, England, 1996.

[23] J. Arlow and I. Neustadt, *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005.

2009 IEEE International Conference on Web Services

Design Quality Analytics of Traceability Enablement in Service-Oriented Solution Design Environment

Liang-Jie Zhang, Zhi-Hong Mao*, and Nianjun Zhou
IBM T.J. Watson Research Center, Hawthorne, NY 10532, USA
E-mail: {zhanglj, jzhou}@us.ibm.com
*Department of Electrical and Computer Engineering, University of Pittsburgh, USA
Email: maozh@engr.pitt.edu

Abstract

This paper provides an artifact-pattern-matching framework and mathematical model to analyze the dynamic behaviors of the SOA solution design in model-driven fashion and provide recommendations for optimal solution pattern enablement for solution artifacts. The artifact-pattern-matching system can be dynamically tuned based on the practitioners' final selections of the recommendations. Specifically, we propose a set of Solution patterns to guide SOA solution architects through the process of consuming and configuring SOA artifacts for composing SOA solutions. The resulting multi-dimensional cascading flagging method is also presented in this paper. As an example, impact analysis patterns are used as solution patterns to support traceability enablement. We present some future directions of leveraging reinforcement learning algorithms to enrich the design quality analytics of SOA solution.

Keywords: Artifact-pattern-matching, SOA, traceability enablement, design quality analytics

1. Introduction

Service-Oriented Architecture (SOA) has been adopted as an extensible architectural framework at the programming and middleware level, business process level and enterprise level for identifying and creating reusable components. Currently, most of the SOA consulting services are conducted in an ad-hoc and labor intensive way due to disparate methods, tools and solution content [1]. The existing methods or techniques for traditional consulting services concentrate on application-specific areas or industry-specific areas. Various documentation based methods or practices have resulted in a challenge of standardizing a design approach for various solution creation scenarios. Especially, in paper-based services delivery practices, it is very hard for solution artifacts created in one scenario to be widely reused or quickly adapted to various types of industry solutions. There is a huge need for improving the traceability of the solution artifacts that are aligned with the business goals to help SOA consultants and architects create a manageable and traceable solution.

In an SOA solution, the solution artifacts include Business Domain, Functional Area, Function, Business Process, Metric, Key Performance Indicator (KPI), Existing Asset, Candidate Service, Composite Service, Service, Service Component, Functional Component, Technical Component, and Component Flow, which are defined in paper [2]. Pattern is a reusable way or template to capture the relationship and attributes of a specific solution artifact or a set of solution artifacts. For example, service allocation pattern is a template for helping a service artifact to be allocated to a corresponding service component artifact, which is to implement the interface and functions designed for the service artifact.

The goal of this paper is to create a systematic way to support practitioners to create and select right solution patterns to apply for selected solution artifacts or models. The right solution patterns can be used as guidance for solution artifacts to comply with industry standards or best practices.

The rest of the paper is organized as follows. Section 2 presents the method of enabling patterns for solution artifacts. Then we formalize the design quality analytics of pattern matching for the solution artifacts in Section 3. We apply pattern matching method to enable traceability among key solution artifacts. The case study is presented in Section 4. The related work is presented in Section 5. Section 6 presents the conclusions and future directions.

2. Method of Enabling Patterns for Solution Artifacts

The artifact-pattern-matching framework includes two major components: Solution patterns, and transformation enablers.

The solution patterns are reusable enablement templates that are used to implement the relationships between SOA artifacts. They are implemented as software components that facilitate required actions based on pre-

978-0-7695-3709-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICWS.2009.145

defined specifications and rules. Thus, individual patterns also record their compliance with guidelines (organizational, industry-specific, or project-specific) that must be followed. In this paper, patterns are implemented to serve as guidance during solution artifacts creation and to build linkages between solution artifacts based on specific requirements. For example, a message specification pattern provides a template for defining the message types, sources, and message formats. The message types include input and output message types. The sources could be local files (e.g. XML schema files) or URLs (e.g. an industry standard based XML schema files hosted on Web sites). The message formats could be string, integer, or complex type. In this case, the message specification is a typical solution pattern that can provide a predefined template for a service to use it to define industry standard compliant messages. From representation perspective, a solution pattern can be captured in UML modeling pattern, XML, or graph.

Transformation Enablers are action items that are trigged when specific context exists or environment is satisfied. In this paper, enablers are realized as menu items in GUI environment and applied as actions to specific solution artifacts in order to facilitate the transformations between solution artifacts.

2.1 Method of creating solution patterns and transformation enablers

The creation process starts with the start step and moves to any phase depending on specific SOA practice's inputs.

The first step is to capture business requirements. Business requirements define the environment within which the solution patterns and transformation enablers will be created.

The next step is to derive SOA specification, which is derived from the business requirements. The specification captures the process and expected results of conducting an SOA practice that fulfills business objectives.

The third step is to identify SOA artifacts. The SOA artifacts are components or products of an SOA practice and identified from SOA specification.

The fourth step is to identify attributes and constraints for SOA artifacts. The attributes and constraints are associated with specific SOA artifacts to describe the unique features of that artifact. Additional artifacts might be derived and vice versa.

Then the next step is to derive solution pattern rules. The solution pattern rules are identified to realize the relationships between artifacts and capture the attributes/constraints of specific artifacts.

The sixth step is to create solution patterns and transformation enablers. After deriving the rules, the solution patterns and transformation enablers can be implemented accordingly.

2.2 Multi-dimensional cascading flagging method

The multi-dimensional cascading flagging method includes solution patterns enablement, transformation enabler activation, pattern selection matching and validation and maintenance of patterns/enablers.

Context-aware pattern enablement

A context-aware pattern enablement algorithm is presented to achieve model specific pattern selection and enablement. The overall architecture of the artifact-pattern-matching framework is shown in Figure 1.

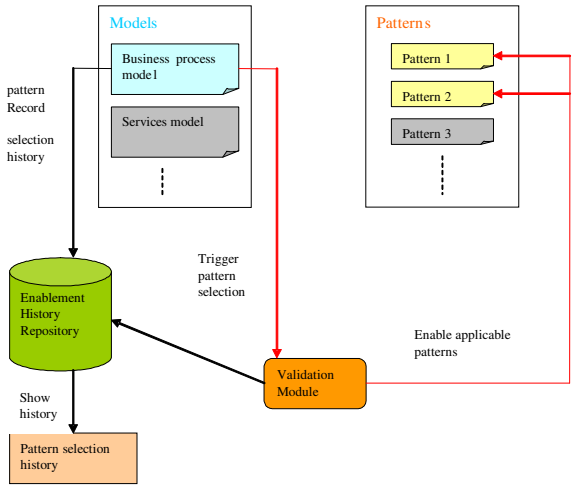


Figure 1. Artifact-pattern-matching framework

This framework includes the following five components:

Models: representing various types of SOA practices such as business process model, services model, etc. Each model consists of a set of solution artifacts introduced in Section 1. When a model or an artifact is selected, an event is triggered to invoke pattern selection algorithm.

Patterns: collecting the set of existing solution patterns. A subset of patterns will be selected to apply to applicable models or artifacts.

Validation module: validating the model/artifact–pattern matching. The validation module takes the selected model/artifact as input, to enable those patterns that are applicable to the specific model/artifact. A model/artifact–pattern look up table is maintained in order to facilitate the selection.

Enablement history repository: recording the pattern enablement history for each model or artifact.

Pattern selection history: displaying the pattern selection history for each model or artifact.

Based on those well-connected components, the execution flow of the pattern selection algorithm is shown in Figure 2.

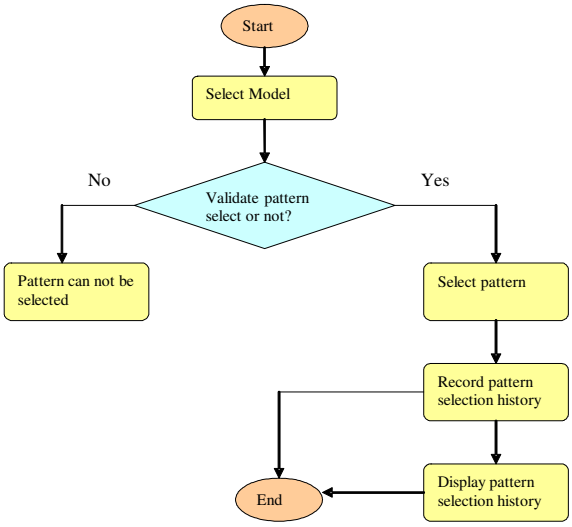


Figure 2. Pattern selection algorithm

After a model or an artifact is selected, the patterns are validated against the selected model/artifact, and only those matching patterns are selected.

The selection history is then recorded as the view of the model/artifact with the pattern applied and can be displayed for later reference. The model-driven pattern enablement provides real-time bindings of patterns and models/artifacts.

It also provides the ability for SOA solution to be quickly constructed. As a result, SOA best practices and guidance can be promoted through systematic linkage between SOA artifacts, attributes, constraints and patterns. Traceability between model and patterns can be enabled by recording and revealing model/artifact-pattern matching history. Finally, valuating a model/artifact against a set of guidelines through pattern-guideline correlation is to ensure compliance to be checked.

Traceability enablement method

An event-driven traceability enablement method is developed to flag SOA artifacts in the following three dimensions.

Flagging artifact-pattern-matching history

As shown in Figure 3, when an architectural artifact is selected, this event triggers the artifact-pattern-matching algorithm. As an example implementation of the algorithm, after looking at the artifact-pattern-matching table, the selected artifact and the associated applicable patterns are recommended and highlighted while the rest artifacts and patterns remain grey as not selectable.

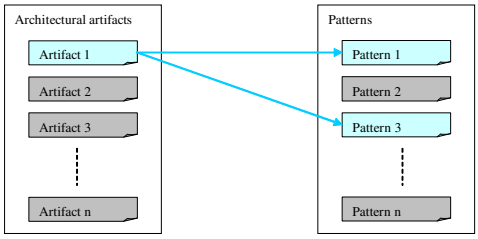


Figure 3. Architectural artifact-pattern flagging

Visual instructions are thus provided to show what the applicable patterns are for a specific architectural artifact. It also improves efficiency and reduces ambiguity during SOA practices when patterns are applied. SOA practitioners can easily choose a pattern which is highlighted to apply to a specific architectural artifact while avoiding the possibility that a wrong pattern might be used.

Flagging sources of architectural artifacts

In Figure 4, geometrical indicators are used to categorize architectural artifacts which are come from various sources. In the SOA solution design environment, there are primarily three categories of architectural artifacts based on where they are derived:

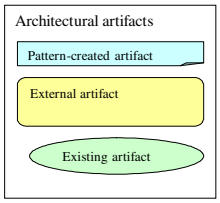


Figure 4. Source flagging of solution artifacts

Pattern-created artifact: it indicates that the architectural artifact is generated by applying patterns.

External artifact: it indicates that the architectural artifact is imported from other resources such as business process analysis results.

Implemented artifact: it indicates that the architectural artifact is an existing SOA component and is currently realized and implemented.

Design artifact: indicates that the architectural element is supplied through a design that has not yet been realized.

Different shapes are used as visualizations of the architectural artifacts from different categories. Additional shapes can be used when other categories are applicable. The shape indicator provides a clear way to show the sources from which an architectural artifact is derived. Thus corresponding analysis steps can be taken for the different categories of the architectural artifacts. For example, impact analysis can be conducted to track the completeness of a design for a solution artifact. Impact analysis engine is one example of transformation enablers

to support direct and indirect impact analysis in SOA solution design context. A case study on this will be presented in Section 4.

The status of an architectural artifact indicates how much work has been done to create or manipulate a specific architectural artifact. A status table is maintained to record the status changes of architectural artifacts. If an architectural artifact's status is changed, the corresponding entry of the status table is updated. A status tag is attached to every architectural artifact to show current status.

Flagging the status of the architectural artifacts

We use percentage to measure the degree of completion. For example, artifact 1 is attached a tag of "20%", that means, 80% to completion. Other measurement scales can be used based on the specific needs of different SOA practices.

The status flagging provides a way to estimate the workload during SOA practices thus a real-time work schedule re-planning can be conducted. Flagging the compliance of the solution to architectural guidelines embedded in solution patterns and transformation enablers.

Since each pattern records its conformance to architectural guidelines, it is possible to verify whether a particular model can be determined to conform to a chosen set of guidelines, by examining the patterns that have been applied to the model, and checking their compliance to selected guidelines using a pattern-guideline correlation table. In this way, the task of certifying compliance is simplified. Additionally, compliance status gives another indication of the progress of the solution.

The above flagging-oriented mechanisms provide the following advantages for SOA practices. First, the non-intrusive flagging mechanism reduces the possible ambiguity in architectural artifact-pattern matching and selection by providing visualized matching and selection method. Second, the source-shape categorizing mechanism improves the productivity in SOA solution design by indicating sources from where the architectural artifacts are derived. Third, the flagging mechanism provides the ability to manage and assess the existing SOA artifact portfolio in the context of design and implementation. Fourth, the status recording and revealing mechanism enhances work planning for SOA practices by estimating the workload for creating or manipulating architectural artifacts. Last, the compliance flagging mechanism allows the SOA practice to monitor compliance to a set of architectural guidelines and estimate remaining workload required to assure compliance.

2.3 Multi-dimensional views of the cascading flagging method

A set of views are provided to give enriched information for SOA practitioners during the realization of the multi-dimensional cascading flagging method. The multi-dimensional view explores the individual artifact dimension, profile dimension, solution dimension and compliance dimension to produce mapping and transformation history.

The individual artifact view

The individual artifact view is triggered when a single artifact history needs to be displayed as in Figure 5.

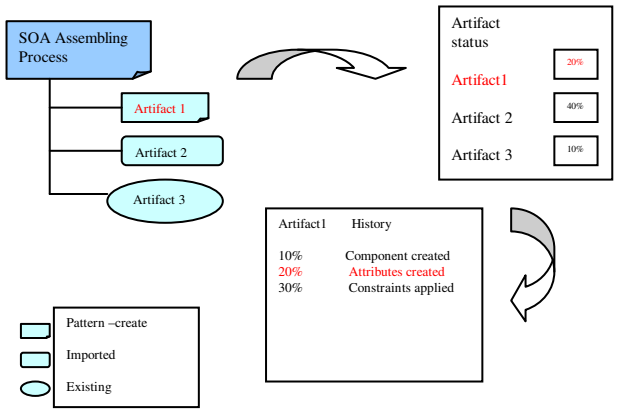


Figure 5: Individual artifact view

For every architectural artifact, its creation status is recorded as percentage and displayed when a specific artifact is selected (highlighted in red color). Upon selecting the status percentage the creation history table is displayed to show the current creation stage of the specific artifact.

The cascading of individual artifact status and creation history view provides a reference for SOA practitioner to estimate the workload of the remaining artifact design. The workload estimation provides linkage to other project management plans. Also, an analysis/change propagation path can be traced to reveal to what extent/stage a change affects an artifact under investigation.

The profile view

The profile view is triggered when a business process is selected as shown in Figure 6. The triggering path is highlighted in red color. When a business process is selected, the corresponding architectural artifacts are displayed. An associated artifact-source table is shown to reveal the sources from where a specific artifact is identified. For every source, a type is attached to show the derivation path of the artifact.

For example, the artifact-source table shows that artifact 1 is created by applying patterns, artifact 2 is imported from results of other business processes and

artifact 3 is an existing asset. For each source, a color tag is used to show the derivation path of an artifact. For example, the yellow means the artifact is newly created, the green means the artifact is transformed from other artifact and the grey means the artifact is reused.

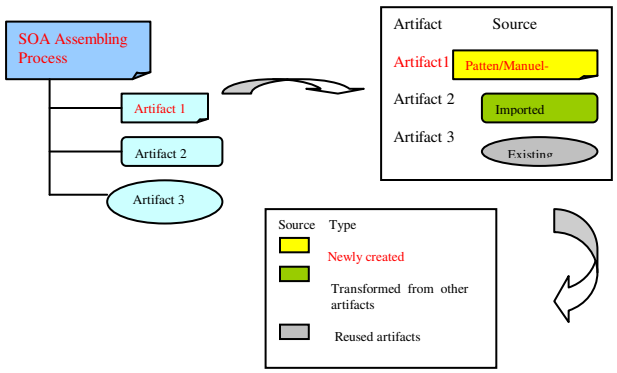


Figure 6: Profile view

The cascading profile view provides a traceable path of artifact creation from source to end-product. Thus, validation can be carried along the creation path to ensure that architectural artifacts are created from the pre-defined source and intermediate transformations.

Compliance view

The compliance view is trigged when an SOA solution's compliance to a set of guidelines needs to be determined as shown in Figure 7.

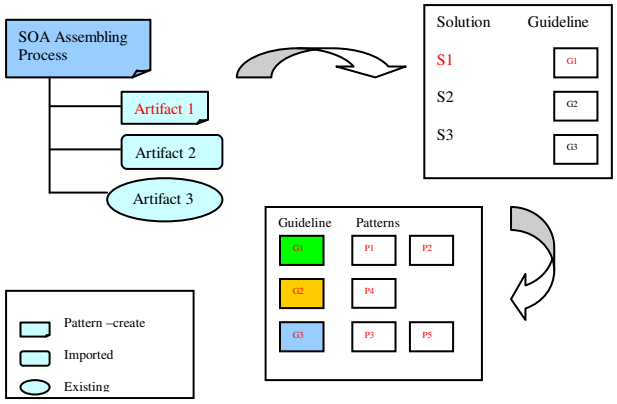


Figure 7. Compliance view

The view depicts the selected set of guidelines that must be followed by the solution, with an associated color for each guideline which indicates the compliance of the solution to that particular guideline. Guidelines for which compliance is verified because a particular pattern or patterns have been applied are highlighted in green color. Those for which compliance has been manually certified by the practitioner are shown in blue. Yellow highlighting is used to indicate guidelines for which the compliance is unknown. For Green and Blue highlighted guidelines, the compliance of the solution can be traced to the individual artifacts which are relevant to the compliance scenarios.

In summary, those multi-dimensional views provide capability of reaching problem definition and determination, visualizing product views and facilitating impact analysis via tracing change propagation path.

3. Design Quality Analytics of Pattern Enablement for Solution Artifacts

Since model is composed by a set of artifacts, we just use artifact as the atomic component of a service-oriented solution. In this paper, we focus on the formalization of the representation of artifacts and patterns, definition of the artifact-pattern-matching, and recommendation of a set of patterns for a selected artifact. This formalization directly impacts the design quality of an SOA solution. That is the reason we use this section to describe the mathematic formalization to cover the design quality analytics of pattern selection for solution artifacts.

Mathematical representations of artifacts and patterns

Denote a_1, a_2, …, a_M the architectural artifacts collected in the current database, where M is the total number of the artifacts. Each artifact can be characterized by a D-dimensional vector

$$a_i = (a_i^1, \cdots, a_i^D), \quad i = 1,...,M$$

where D is the number of attributes of the artifact and each element in the above vector, a_i^l, l = 1,…, D, is a scalar quantifying the l-th attribute of artifact a_i.

Denote p_1, p_2, …, p_N the patterns collected in the database with N being the total number of the patterns. Each pattern can be characterized by a graph: $p_k = \{V_k, E_k\}$, $k = 1,...,N$.

In the above expression, V_k is the set of vertices in the graph, and is further denoted $\{v_k^1, \cdots, v_k^{m_k}\}$, where each element or vertex represents an artifact and m_k is the total number of artifacts in V_k; E_k is the set of edges in the graph, which contains a total of n_k edges, $\{e_k^1, \cdots, e_k^{n_k}\}$, and each edge represents a relationship or connection between a pair of artifacts.

Similarity between two artifacts

Since later we need to quantify the similarity between two artifacts, here we introduce a similarity measure for two artifacts (vectors) $a_i = (a_i^1, \cdots, a_i^D)$ and

$a_j = (a_j^1, \cdots, a_j^D)$:

$$s(a_i, a_j) = e^{-\frac{\|a_i - a_j\|}{\sigma}} \quad (1)$$

where $\|a_i - a_j\| \equiv \sqrt{\sum_{l=1}^{D} (a_i^l - a_j^l)^2}$ and σ is a positive coefficient. Note that the value of similarity ranges from 0 to 1. The smaller the difference between the two vectors is, the larger the similarity measure is. The similarity takes the largest value, i.e. 1, when the two vectors are exactly the same. The coefficient σ indicates the sensitivity of the similarity measure with respect to the absolute difference between the two vectors: The smaller σ is, the faster $s(a, v)$ approaches zero as the difference between *a* and *v* gets larger.

Applicability of a pattern to an artifact

It is desirable that we quantify the relevance between an artifact and a pattern or the likelihood with which the pattern is applicable to the given artifact. This would allow us to select appropriate patterns from the database to guide the usage of artifacts. We use $r(a_i, p_k)$ to represent such a likelihood with which pattern p_k is applicable to artifact a_i. The value of $r(a_i, p_k)$ ranges from 0 to 1. In the following, we show how $r(a_i, p_k)$ is calculated and how it can be learned from expert evaluations.

We begin with the earliest stage of creating the database when we have a set of *M* artifacts and *N* patterns but have not initialized the values of $r(\cdot, \cdot)$ yet. Consider a specific pair of artifact a_i and pattern p_k. Obviously, if a_i belongs to $V_k = \{v_k^1, \cdots, v_k^{m_k}\}$, then $r(a_i, p_k)$ should equal 1. This is because a_i is already an artifact in the vertex set V_k of pattern p_k and thus p_k has to be applicable to a_i (with the highest likelihood 1). It is natural for us to further assume: If a_i is "close" or "similar" to one of the artifacts in V_k, then $r(a_i, p_k)$ should return a big value, i.e., pattern p_k is likely to be applicable to a_i. Therefore, we may set the initial value of $r(a_i, p_k)$ based on the similarity between a_i and its closest artifact in V_k:

$$r(a_i, p_k) = \max_{j=1,\ldots,m_k} \{s(a_i, v_k^j)\} = \max_{j=1,\ldots,m_k} \left\{ \exp\left(-\frac{\|a_i - v_k^j\|}{\sigma} \right) \right\} \quad (2)$$

The above initial estimate of $r(a_i, p_k)$ can be refined when expert evaluations are available. Assume that we have the opinion from an expert, who estimates the value of $r(a_i, p_k)$ to be $r_{\text{expert}}(a_i, p_k)$. Then $r(a_i, p_k)$ can be updated by

$$r_{\text{new}}(a_i, p_k) = r(a_i, p_k) + \eta_{\text{expert}} \left[r_{\text{expert}}(a_i, p_k) - r(a_i, p_k) \right] \quad (3)$$

where $\eta_{\text{expert}} \in [0,1]$ is an adaptation coefficient. A bigger value of η_{expert} implies a higher level of expertise of the expert. When we can fully trust the expert, we set $\eta_{\text{expert}} = 1$ and using (3) we obtain $r_{\text{new}}(a_i, p_k) = r_{\text{expert}}(a_i, p_k)$. If more than one expert evaluations are available, we can repeat the calculation in (3) iteratively [note that in each iteration we need to replace $r(a_i, p_k)$ with the last calculated $r_{\text{new}}(a_i, p_k)$].

When a new artifact, say a_{M+1}, is added to the database of *M* artifacts, we can predict its potential relevance with an existing pattern in the database, say p_k, using a formula similar to (2):

$$r(a_{M+1}, p_k) = \max_{j=1,\ldots,m_k} \{s(a_{M+1}, v_k^j)\}. \quad (4)$$

Or we may estimate $r(a_{M+1}, p_k)$ based on the prior knowledge about the applicability of p_k to the existing artifacts in the database:

$$r(a_{M+1}, p_k) = \frac{\sum_{i=1}^{M} s(a_{M+1}, a_i) r(a_i, p_k)}{\sum_{i=1}^{M} s(a_{M+1}, a_i)}. \quad (5)$$

Formula (5) simply is a weighted sum of $r(a_i, p_k)$ for all a_i in the database. The weights are determined by the similarities between a_i and a_{M+1}. Whether to use (4) or (5) depends on the relative reliability of the two methods. When expert evaluations are available, $r(a_{M+1}, p_k)$ can be updated using formula (3).

4. Case Study: Traceability Enablement through Domain-Specific Impact Analysis Patterns

In this section, we give a case study of applying the method developed to define an impact analysis in an SOA design environment. For an architect, such impact analysis is a UI display of certain portion of the design related to a specific artifact under investigation. Such analysis is very important for a variation-oriented design.

Mathematically, an SOA design can be represented as a graph with nodes presenting artifacts and edges (or links) representing relationships among nodes. Therefore, the impact analysis of an artifact (say ***i***) is mathematically defined as to compute a sub-graph of a design using artifact ***i*** as input. Unfortunately, a design model for a real industry application could be very large and complex

with several thousand artifacts. The impact of an artifact could be complex when considering the direct and indirect relationships through other artifacts. The engineering challenge is to select only those artifacts and relationships that are important to the artifact to be analyzed. We will discuss how to solve this problem through learning from historical practices from different architects.

Here we use our SOA modeling practices to illustrate the potential complexity of such impact analysis could be and to formulate a solution. We use different tools at different design phases to capture architectural artifacts in different granularity. First, we use an enterprise modeling tool and business process modeling tool (such as IBM WebSphere Business Modeler) to capture high-level enterprise components and business processes. Second, we use a service model (such as SOMA-ME service model) to perform service identification and specification. Third, once all those services are identified and specified, we can use an reference architecture model (such as SOMA-ME S3 model) for architecture design. Finally, such architecture design is mapped into service component architecture (SCA) as deployable artifacts (BPEL, WSDL, and XML or database schema). Those models in different phases are coupled together through the artifact links of the models (in UML, relationships are represented as dependency, association, or containment).

The impact of an artifact can be defined as direct and indirect. Mathematically, a direct (indirect) impact diagram of an artifact ***i*** is a graph contains only those artifacts (and corresponding links) that can be traced back to (from) ***i*** with ***i*** as the only root (leaf) node. The direct impact analysis is used to track the implementation of an artifact, which can tell how the artifact is progressively realized. The indirect impact analysis is used to find out how an artifact is identified from and aligned with business requirements through bottom-up analysis.

To simplify the problem of identifying the scope of an impact analysis, we utilize the meta-model concept. In an SOA design, an artifact has to have a type. The type of an artifact determines what possible relationships with other artifacts are. Such type definition and relationship constrains are defined as meta-model, and represented as a meta-graph [4]. We assume that the scope of impact analysis is only determined by the type of an artifact (say ***i***). Such scope is defined by a set of artifact types and relationship types represented as sub-graphs of the meta-model as a function of the artifact type of ***i*** .

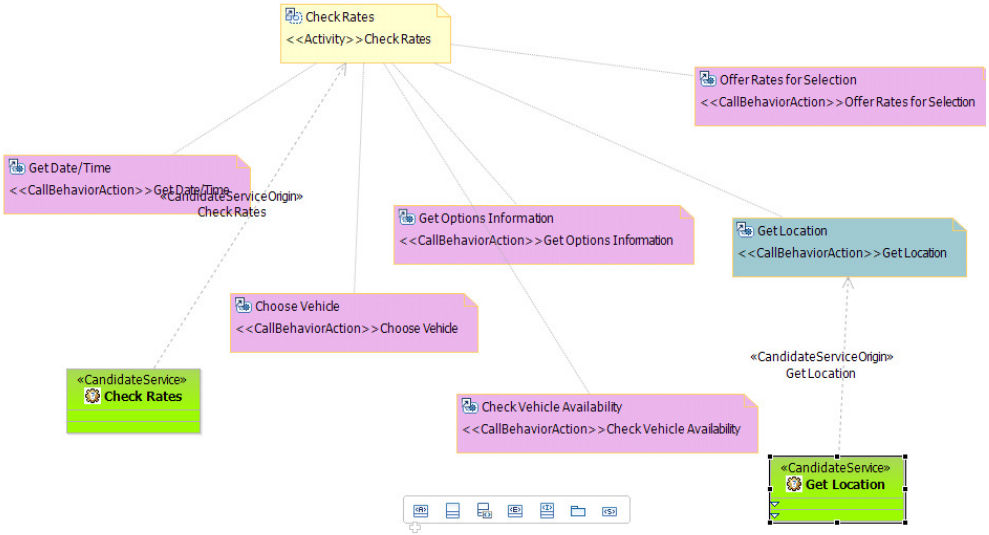


Figure 8. Direct Impact Analysis - from Business Process Model to SOMA-ME's Service Model

As we only consider the artifact type in this analysis, we define the distance and similarity of two artifacts as follows. If two artifacts (a_1 and a_2) have same artifact type, we have $\|a_i - a_j\| = 0$ and $s(a_i, a_j) = 1$. Otherwise, we have $\|a_i - a_j\| = \infty$ and $s(a_i, a_j) = 0$. The learning process starts with collecting the engineering practices of impact analysis from different architects/practitioners. Such practices (represented as impact diagrams) are abstracted as sub-graphs (of mete-model) stored into the knowledge database discussed in Section 2. Finally, we have the most suitable impact analysis scope through the learning for each artifact type.

We enable such impact analysis in SOMA-ME tooling environment through automatically generating impact analysis diagrams. Figure 8 illustrates a direct impact diagram of a business activity "CheckRates" cross business process and service identification phases. In impact diagrams, we apply certain color coding to warn an architect those artifacts might have problems or require attention in the design.

5. Related Work

Along with objected-oriented design and design patterns, architecture styles have been realized as a very important [5] and promising approach to simplify software design and reuse by capturing and exploiting system design knowledge. In the efforts of Monroe et. al., the authors explore the capabilities and roles of the architecture styles approach, its strengths, and its limitations. In our paper, we propose a set of consumption and configuration patterns to guide architects for composing SOA solutions.

Quality-driven architecture design and quality analysis (QADA) [6] is an approach to software architecture design which emphasizes the importance of addressing quality attributes. The approach treats quality attributes same important requirements as functional requirements and constraints. While their work focuses on macro aspect and methodology level of design quality, our work focuses on how to capture the design quality patterns from engineering practices through a learning process. These two efforts can complement with each other to achieve better quality design.

Sartipi and Kontogiannis proposed a graph pattern matching approach to software architecture recovery [7]. Different from their effort of analyzing the pattern from existing legacy systems, our matching algorithm focuses on the pattern matching from engineering practices for architecture designs. The other difference is that we are using a different matching algorithm by defining the attribute similarity and pattern learning process.

Zdun and Dustdar proposed process-driven SOA models via a model-driven software development approach based on software patterns [6]. They introduced pattern primitives as an intermediate abstraction to formally model the participants in the solutions that patterns convey. In our research, we do not focus on the pattern primitives, but how the pattern primitives can compose a quality solution through the accumulated practice experience through learning process.

6. Conclusions

In this paper, we have presented a method of creating and managing solution patterns and transformation enablers to consume and configure architectural artifacts for SOA solution design. We have also formalized a mathematical model for recommending or selecting right patterns to apply for selected solution artifacts. As a case study, we have presented impact analysis as an example solution pattern. We have also used direct impact analysis and indirect impact analysis to show solution rules and the corresponding impact analysis engine to guide the practitioners to apply right patterns to the solution artifacts in the solution design lifecycle.

In the future work, we would like to work in parallel on the following two directions. First, we will collect more practices from the experienced practitioners on solution patterns and usages of those patterns to build a solution pattern database for SOA design. On the other hand, we will leverage or enhance some learning algorithms (e.g. reinforcement learning) to adaptively tune the artifact-pattern-matching system to analyze the dynamic behaviors of the SOA solution design environment based on newly available best practices.

7. Acknowledgement

We would like to send our thanks to Dingding Lu who was a summer intern at IBM in 2006. She participated in the discussions on some of the initial ideas presented in this paper.

8. References

[1] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*, Springer and Tsinghua University Press, 2007.

[2] L.-J. Zhang, N. Zhou, Y.-M. Chee, A. Jalaldeen, K. Ponnalagu, R. R. Sindhgatta, A. Arsanjani, and F. Bernardini, "SOMA-ME: A Platform for the Model-Driven Design of SOA Solutions," IBM Systems Journal, vol. 47, no. 3, pp. 397-413, 2008.

[3] L.-J. Zhang, A. Arsanjani, A. Allam, D. Lu and Y-M Chee, "Variation-Oriented Analysis for SOA Solution Design," Proc. Int'l Conf. Services Computing (SCC'07), pp. 560-568, 2007.

[4] N. Zhou, Y.-M.. Chee, L.-J.Zhang: Coding-Free Model-Driven Enablement Framework and Engineering Practices of a Context-Aware SOA Modeling Environment. IEEE ICWS 2008: 553-560

[5] Monroe, R.T. Kompanek, A. Melton, R. and Garlan, D. "Architectural styles, design patterns, and objects," Software, IEEE: Jan/Feb 1997, Vol: 14, Issue: 1, pp 43-52

[6] M Matinlassi, E Niemelä, L Dobrica, "Quality-driven architecture design and quality analysis method", VTT Publications. ISBN 951-38-5967-3

[7] Kamran Sartipi, Kostas Kontogiannis, "A Graph Pattern Matching Approach to Software Architecture Recovery," ICSM, pp.408, 17th IEEE International Conference on Software Maintenance (ICSM'01), 2001

[8] Zdun, U. and Dustdar, S. "Model-driven and pattern-based integration of process-driven SOA models", International Journal of Business Process Integration and Management, Vol. 2, Num. 2, 2007, pp109 – 119

2008 IEEE International Conference on Services Computing

A Graph Theory Based Impact and Completion Analysis Framework and Applications for Modeling SOA Solution Components

Nianjun Zhou, Liang-Jie Zhang
IBM T.J. Watson Research Center
Hawthorne, NY 10532, USA
{jzhou,zhanglj}@us.ibm.com

Abstract

Evolved from our engineering experience, this paper presents a mathematical framework to define and analyze an SOA (Service-Oriented Architecture) model. SOA model, composed by design elements, is represented as directed graph based on graph theory. For each design element, two directed graphs are created to reflect the panoramic view and relationships of this design element with other design elements of same model, and used for impact and completion analysis of this design element. A numerical value called the relative importance indicator is computed to quantify the relationship between any two design elements. This indicator forms a matrix that is used as a base for more advanced analysis, such as model partition, model coupling, and variation-oriented design. Some future research directions such as model reduction are discussed at the end of this paper.

Keywords: SOA, modeling, relative importance indicator, model partition, graph theory, impact and completion analysis, variation-oriented design

1. Introduction

Service-Oriented Architecture (SOA) provides a loosely coupled architectural style to conceptually support solution design that leverages various design elements such as business processes, services, packaged applications, and data throughout its design, build and manage lifecycle. SOA design covers enterprise operations to satisfy the business goals of an enterprise. From bridging the gap of business and IT perspective, SOA also helps define and provision IT infrastructure to realize and support different applications to exchange data in business processes, which are independent of the operating systems and programming languages underlying those applications. SOA is being adopted as a popular open architecture for business solutions [1]. The high-level SOA design idea can leverage model-driven design (MDD) to create a consistent and systematic way of supporting application development which focuses on connecting business goals, service identification, service specification [2] and service realization.

An SOA solution contains multiple design and implementation phases in its whole lifecycle. Many conflicted factors have to be considered, such as performance over cost saving. Usually, a solution design is modeled in model language UML (Unified Modeling Language (UML) [3] or descriptive languages, such as XML[7]. Often, the business goals, functional requirements and architecture decisions change during an SOA solution development. Therefore, there are constant needs to analyze and check the model integrity. These needs come from 1) model validation – where we need to valid that model following certain design patterns and design logic; 2) variation-oriented analysis and design – where it allow us to have quick adapt to the change of design requirements; 3) relationship-based design elements discovery and composition – where we can discover the required services and composite new service; 4) model structure analysis – where it allow us to analyze the structure of a business application and related service quality; and 5) predication of runtime monitoring and management based on design element impact analysis. The support of these features is a necessity of creating a software-based design platform for SOA practices. All these requirements need a quick view of a model in a global perspective rather than focusing on a specific building block (design element).

We formulate our SOA design to create an analytic model to support SOA design. Our framework first abstracts modeling methodology into a *meta-graph*; then abstracts an SOA model as an *instance-graph* annotated by the *meta-graph*. Mathematical algorithms are applied to analyze the *instance-graph* for integrity check, to identify possible faults and defects in earlier stages of a design, and to provide certain engineering indicators to serve a design architect. The introduction of mathematic rigorousness is our attempt to shift the IT architecture design from artistic work to more formalized engineering practice to help speed up SOA adoption.

The paper is organized as follows. Section 2 provides an overview of our engineering effort of creating SOA modeling environment and carrying on impact analysis. Section 3 presents the mathematical formulation and introduces a relative importance matrix to quantify the impacts of design element. It also provides the model integrity check algorithm and some engineering application of the introduced matrix. Section 4 discusses the related work. Section 5 summarizes the results and points out some possible future directions.

978-0-7695-3283-7/08 $25.00 © 2008 IEEE
DOI 10.1109/SCC.2008.137

2. Challenges and Solution of SOA Modeling Integrity Check

2.1. Overview of Our SOA Modeling Environment

We start from a brief introduction of our SOA modeling process, which contains the following three phases [2][4]:

- Service Identification – to identify *candidate services* using techniques of domain decomposition, goal-service modeling and existing asset analysis;
- Service Specification – to specify the details of service requirements, service dependences, component model as well as the logical data model;
- Service Realization – to realize services, subsystems, functional and technical components.

Figure 1 shows the high level overview of such SOA model. Each phase is represented inside a box. Within the box, certain modeling tasks have to be completed. For service identification, a design architect has to start from *business goal*, *business process*, and *application domain*, and step-by-step to find out *candidate service(s)*.

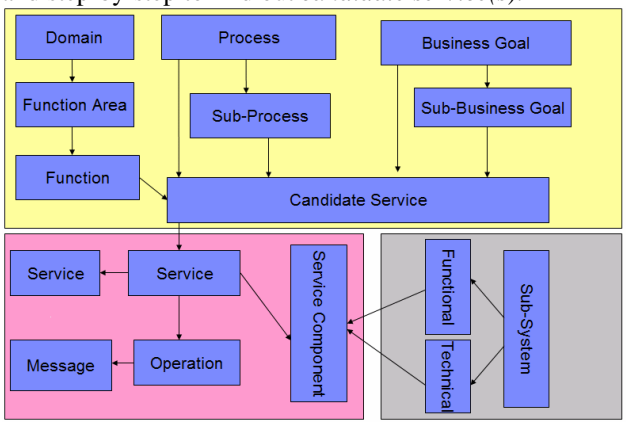


Figure 1. SOA-ME Stereotypes and Relationship

Our SOA modeling environment (SOMA-ME) [2][4] prototype is built upon UML [3] (as modeling language) and IBM Rational Software Architect (RSA) (as development platform). UML Profile is used to define the stereotypes of design elements. From Figure 1, the stereotypes for service identification phase are *business goal*, *process*, and *domain, function area, function and candidate services*. We allow *business goal* (*process*) to include another *business goal* (*process*).

In SOMA-ME prototype, we created an UI environment with a set of context-aware menu to guide our design architects. Therefore, a design architect does not to use UML directly to create SOA model. This helps to eliminate certain errors due to misuse of the UML and to reduce the barriers of creating of an SOA model.

2.2. Model Impact Analysis and Integrity

One of the key feedbacks of using SOMA-ME prototype is the need of performing model validation and completeness check. For example, we need to check 1) certain *business goal* is fully supported with proper identified *services*; and 2) no redundant *service* or *service components* are created. We cannot rely on individuals' eyes and brains to remember all the details of a modeling effort. In our SOMA-ME, we add the impact analysis feature to provide design architect the views of the impact of a design element through direct and indirect impact diagrams.

For a selected design element (*a*), direct impact diagram displays those design elements (and their relationships) can be traced back to *a* (Figure 2). The traceability is enabled using existed UML relationships, such as folder containment, UML dependency/association relationship. We call those design elements as offspring of *a*. In Figure 2, from the *business goal* "Increase Revenue by 20% by the end of Fiscal Year 2008" (design element *a*), a design architect creates three *sub-business goals*, and further identifies other design elements *candidate* service(s), *service(s)* and *service components* following certain design steps. Conceptually, when design element *a* is removed (or invalid) from the design model, those offspring design elements become either invalid (if it is deduced only from *a*) or questionable (if it can be deduced from other design element(s) besides *a*). In Figure 2, some of the *sub-business goals* and *candidate service* are marked as pink color. This color is used to alarm a design architect that these design elements have problem. Pink colored design elements should not be the ending design elements in a design, should have other design elements deduced from them. For example, *a (sub-) business goal* should be used for identifying *candidate service(s)*. Therefore, a *(sub-) business goal* that is not followed by any *candidate service* design element is questionable and marked as pink.

Similarly, for a selected design element (say *b*), indirect impact diagram displays those design elements (and their relationships) can be traced from *b* (Figure 3). We call those design elements as ancestors of *b* (viewed as the design elements where the selected design element *b* deduced from). In Figure 3, a *service component* is created for hosting *services* of *cancel reservation*, *check-in vehicle*, and *check-out vehicle*. Again, conceptually, when the selected design element - *b* (say *service component*) is removed from the design model, those ancestor design elements become either invalid (if it can be only traced from design element *b*) or questionable (if it can be traced from other design element besides *b*).

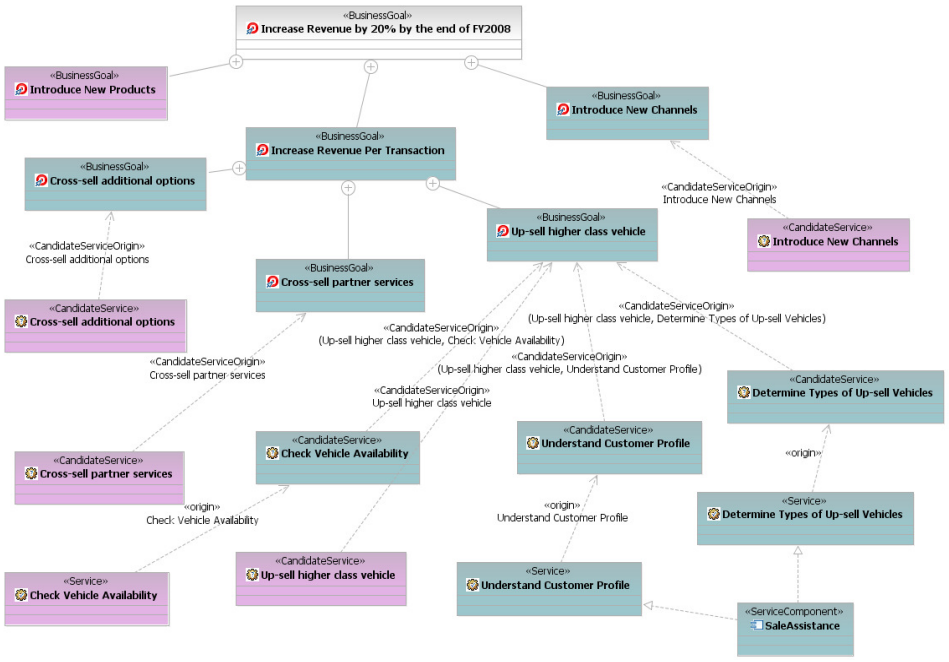


Figure 2. Direct Impact Diagram for a *Business Goal* Design element for an End-to-End Impact Analysis

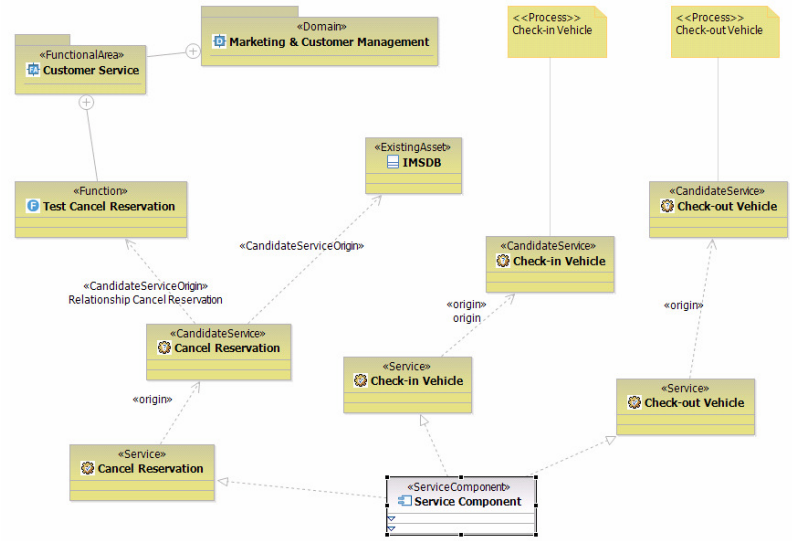


Figure 3. Indirect Impact Diagram for a *Service Component* Design Element

This impact analysis feature is implemented as a plug-in to IBM RSA platform contains configuration file. The file includes the information of:

- The legitimate UML stereotypes in a model;
- The legitimate UML stereotypes a model should start from and end at;
- The legitimate UML relationship type (UML dependency stereotype) is allowed for two design elements with given stereotypes;
- The legitimate location in the model for a design element with given stereotype;

To accommodate the needs of having different scopes of impact analysis, we allow multiple configuration files to be used. For example, "Identification Phase Impact Analysis" is corresponding to impact analysis for *service identification* phase; "End-to-End Impact analysis" is corresponding to end-to-end impact analysis for all three design phases.

In Section 3, we will continue discuss the direct (or indirect) impact diagrams, and abstract them as a directed

graphs with the selected design element *a* as the only root node[1] (or leaf node).

3. Proposed Mathematical Framework

In this section, we introduce the necessary mathematic abstraction and definition to map both meta and design models into graph representations. The model integrity validation and impact analysis algorithms are created by traversing the design model level graph. We introduce the relative importance indicator matrix, and some engineering indicators deduced from this matrix.

Before defining our abstraction, we like to point out the difficulties of performing model integrity check and analysis today in current engineering practices partially come from lacking such abstraction and constraints. A regular UML modeling tool allows a design architect to have the freedom of:

- adding design elements that are not conformed with well-defined stereotypes;
- adding design element linkage, folder containment relationship of any two design elements;

Such freedom limit us to justify whether a design model 1) has followed proper development steps; 2) has created without any duplicated design elements or extra relationships; and 3) has missed any necessary design elements in the model.

3.1. SOA Model Abstraction

3.1.1. Meta-Model Abstraction

A *meta-model* is consisted of a collection of *design element stereotypes* (called *stereotypes* thereafter) and the *associations of stereotypes*. The abstraction is presented as a directed graph (called *meta-graph* thereafter) consisted of nodes (*stereotype*) and edges (*associations*). The *association* is relationship starting from one stereotype and ending at another stereotype, representing a logic step of an SOA solution design. For example, a *business goal* is used to identify *candidate service*(s). Therefore, an edge (*associations*) exists from stereotype of *business goal* to stereotype of *candidate service.* We allow an *association* be used to link back to same *stereotype*. Such self-pointed *association* is used to capture the following two scenarios: 1) recursive relationship existed in *design elements* with certain *stereotypes* (for example, a *business goal* can have sub-*business goal*, a *service component* may contain other *service component(s)*); and 2) dependency relationship, such as *service* dependencies.

Without considering those self-pointed edges (*associations)*, we constrain a *meta-graph* must be acyclic. The rational behind this is two-folder. First, an SOA solution design usually follows certain logic steps. Capturing such steps into a *meta-graph* will naturally form an acyclic graph. Second, design elements of a model usually are organized into certain hierarchical structure. Such structure also forces us to have an acyclic representation.

We believe that each SOA model needs to have a well-defined design scope. Therefore, we introduce the ideas of *starting stereotype* and *ending stereotype* as constraints for limiting where a design model could start from and end at. In Figure 1, the *starting stereotypes* are *business goal*, *process*, and *domain.* The *ending stereotype* is *service component*.

Furthermore, we apply two more constraints to the *meta-graph*.

- Connected Graph - The rational is that a model logic concepts/steps have to be an integrated piece, and cannot be separated into multiple isolated pieces;
- Uniqueness of *Association* – between two *stereotypes*, at most one *association* is allowed to exist. The rational for such constraint is that we want to make sure that each *association* represents an atomic logic step of a modeling process. Therefore, any possible ambiguous should be eliminated.

For any edge (*association*) defined, we define two attributes for model integrity check. They are

- A logic variable that defines the necessity of having such relationship in an instance of design model.
- An enumeration variable to define the cardinality allowed for this relationship. The possible values of the variable are *'1', '*', '1…*', '0…1', '(0,1)', '(1,n)', '(0,n)'*, and '*(1,1)*'.

Finally, we can use '$\succ$' to represent a partial order relationship for *stereotypes* (nodes). For any two stereotypes of s_a and s_b, we say $s_a \succ s_b$ if there is a path from s_a to s_b in the meta-graph.

3.1.2. Design Model Abstraction

A *design model* is represented by an *instance-graph* consisted of design elements as graph nodes and links as directed edges, where each design element/link belongs to ONLY one valid *stereotype/association* in the *meta-graph*. An *instant-graph* is annotated by a *meta-graph*.

[1] Figures of 2 and 3 are generated using IBM RSA tool. In our formal mathematical definition in Section 3, the arrow direction always follows the logic step of a design. Therefore some arrows could be just opposite as shown in Figures of 2 and 3.

Each design element/link has a unique identifier. In the context of the model-driven architecture, a link is an abstraction of various relationships in the model - including folder containing, design element dependency, and usage relationship.

An *instance-graph* is a *directed acyclic graph*[2]. Similarly, we can define a partial order relationship '$\rhd$' for design elements. For any two nodes (design elements) of a_a and a_b, we say $a_a \rhd a_b$ if there is a path from a_a to a_b in an *instance-graph*. We have:

- If $a_a \rhd a_b$ and $a_b \rhd a_c$, then $a_a \rhd a_c$;
- If $a_a \rhd a_b$, then $s(a_a) \succ s(a_b)$. Here $s(a_a)$ and $s(a_b)$ are the *stereotypes* of a_a and a_b. This property connects two partial orders ('$\succ$' and '$\rhd$') defined in *meta-* and *instance- graphs*.

In short, the introduction of *meta-graph* and *instance–graph* allow us to add SOA design specific semantics into graph node and edge in traditional graph theory. Such difference becomes the base for model integrity check.

3.2. Mathematical Representation of Model and Design Element Integrity

A design model consistence with the constraints defined in Section 3.1.1 is called a *valid design model*. The constraints are applied to *stereotype* and *association* definition and *cardinality* requirement.

3.2.1. Design Element Integrity

In Section 2, we introduce the ideas of direct and indirect impact diagrams to reflect the relationship of a selected design element (say *a*) with other design elements. Using the *instance-graph*, such direct/indirect impact diagram becomes a connected directed acyclic sub-graph of the *instance-graph* with *a* as the ONLY root/leaf node. We call it as *direct/indirect impact-graph*. The union of the two graphs is also a sub-graph of the *instance-graph* (called *join impact-graph* for *a* thereafter), and reflects the portion of design related to *a*. In our context, we define a design element is *valid* if the *joint impact-graph* itself is a *valid design model*. Although an *instance-graph* may not need to be connected, a *joint impact-graph* has to be connected.

Now, we are ready to define integrity for a design element. First, from the *direct impact-graph*, if some of the leaf nodes (i.e. *ending design elements*) do not belong to *ending stereotype(s)*, the design element (*a*) is defined *partial incomplete* design element. If none of its leaf nodes belong to *ending stereotype(s)*, the design element (*a*) is called *fully incomplete* design element. Similarly, from the *indirect impact-graph*, if some of the root nodes (i.e. *starting design elements*) do not belong to *starting stereotype(s)*, the design element *a* is defined *partial isolated* design element. If none of its root nodes belong to *starting stereotype(s)*, the design element is called *fully isolated* design element. The discovery of such *incomplete* or *isolated* design element(s) becomes the base for integrity check and color coding discussed in Section 2.

In the context of this paper, the definition of validness of a model or a design element is purely based on the graphic structure and constraints in Section 3.1 (including acyclic and cardinality constraints). These are the minimum requirements we found to define validness of a model. Other requirements/rules could be introduced based on the semantics of a specific design model, which is beyond the scope of this paper.

3.2.2. Model and Design Element Integrity Check Algorithm

Now, we are ready to create the algorithm for model integrity check for a design model (*instance-graph*):

- Loop all the root nodes (design elements) associated with *starting stereotypes*;
- For each root node (design element), generates corresponding *direct impact-graph*;
- Check the edges in the *direct impact-graph* satisfied the stereotype and cardinality constraints defined in *meta-graph*;
- Check the leaf nodes (the ending design elements) of each *direct impact-graph* associated with *ending stereotypes* defined in *meta-graph*;
- Check each node (design element) in the *instance-graph* to make sure it appeared at least once in the generated *direct impact-graphs*.

A model is valid if it passes the above validation check. For design element integrity check, we first generate the *join impact-graph* for a design element. Then verify the *join impact-graph* is valid using same algorithm (omit last step).

3.2.3. Discovery Algorithm for Isolated Design Element

Changes in design usually result in loss of certain traceability/connectivity of some design element(s) with other design element(s). Then those design elements become useless parts in the design model. It is similar to programming practices, where some code elements in the program could be useless and not be reached in execution time. To discover those isolated design elements, we can use the same algorithm for model integrity check but replacing the last step as:

[2] From our engineering experience, cyclic loop in design model is one of the major reasons creating bugs when a model is used to automatically generate programming level scripts or codes.

- If a node (design element) in the *instance-graph* does not appear in any generated *direct impact graph* from root nodes, it is a *full isolated design element*.
- For each *full isolated design element* (say *a*), we generate *direct impact-graph* for this *full isolated design element a*. Those design elements (has *a* as their ancestor) in the generated *direct impact-graph* are at least *partial isolated design elements* in the design model.

3.3. Quantifying the Relative Importance of An Node (Design element)

We define a *relative importance indicator* to quantify the importance of one design element (node) over another. Before our discussion, we assign an index as identifier to each design element (node) using nature number (1,2,3,…). *Instance-graph*, *as directed acyclic graph*, has one or more topological sorts [5] due to the existence of the partial order ($\triangleright$) property among the nodes. We can choose one of the topological sorts as the base for creating such identifier.

We define the relative importance of two design elements (say i and j) is zero if there is no link path in the *instance-graph*. We only need to discuss when i and j have a path ($i \triangleright j$).

3.3.1. Relative Importance Indicator from an Ancestor Node to its Offspring Node

CASE 1: There is a directed edge/link from *i* to *j*. It is possible that there are other design elements also having directed edges pointing to design element *j*. Together with *i*, these design elements form a subset (say *A* with cardinality as *m*). Then, we define the relative importance indictor as $x_{i,j} = 1/m$ (i over j). To introduce some semantics into relative importance indicator without adding too much complexity, we can give different weights (positive values) over edges based its corresponding *association* in *meta-graph w(s(i),s(j))*. We use *s(x)* to present the stereotype of a design element *x*. We can define the relative importance indicator of $x_{i,j}$ as

$$x_{i,j} = \frac{w(s(i), s(j))}{\sum_{a \in A} w(s(a), s(j))} \qquad (1)$$

There is an engineering reason for such weighting. For example, a *candidate service* is identified through two paths: 1) from *business goal*, and 2) *domain→ functional area→function*. We might want to give more weight for the second path with the consideration that service identified through *domain→ functional area→function* might be more accurate than through *business goal*, as *business goal* usually is fuzzier in the technology area.

CASE 2: Assume there is a path from i to j (i is an ancestor of j), but linked through other intermediate nodes (design elements), we compute relative importance indicator $x_{i,j}$ as follows:

1. Find out all the paths (forms set *P*) from i to j in *instance-graph* (some paths might have portion overlap, but not total overlap);
2. For each path of $p \in P$, we multiply the relative importance indicator for each edge in path p and have a value for each path, say $x(p)$;
3. The relative importance indicator of $x_{i,j}$ is computed as the summation of $x(p)$ over all $p \in P$.

In the case of $j \triangleright i$, we define $x_{i,j} = 0$. If i and j are not connected, we define $x_{i,j} = x_{j,i} = 0$. If i and j are identical, then $x_{i,i} = 1$.

3.3.2. Relative Importance Indicator from an Offspring Node to its Ancestor Node

The process of computing the relative importance indictor is almost identical as Section 3.3.1. We choose $w(s_a, s_b) = w(s_b, s_a)$. s_a and s_b are two stereotypes.

CASE 1: There is a directed edge from i to j. We define

$$y_{j,i} = \frac{w(s(i), s(j))}{\sum_{a \in A} w(s(i), s(a))} \qquad (2)$$

Set *A* (with cardinality *n*) contains the child nodes (design elements) of i. If i only has j as its child, then $y_{j,i} = 1$. In the special case of treating all edges equal, we have $y_{j,i} = 1/n$.

CASE 2: Assume there is a path from i to j (i is an ancestor of j), but linked through other intermediate nodes (design elements). We compute relative importance indicator $y_{j,i}$ same as CASE 2 of Section 3.3.1 except that we use $y_{j,i}$ to replace $x_{i,j}$.

In the case of $j \triangleright i$, we define $y_{j,i} = 0$. If i and j are not connected, we define $y_{i,j} = y_{j,i} = 0$. If i and j is identical, then $y_{j,j} = 1$.

Lemma 1: Let *E* be the set of all leaf nodes (*ending design elements*) of a *direct impact-graph* for node (design element) *i*. We have:

$$\sum_{e \in E} y_{e,i} = 1 \quad (3)$$

Proof. If *i* itself is a leaf node, then $y_{i,i} = 1$ by definition. Let us denote *l* as the length of a longest path in the *direct impact-graph*. We prove the lemma over the value of l recursively. First, if *l=1*, from definition (2), we know the result in (3) holds. Now, assume that Lemma 1 is true for all $l \le n$ (n as a positive integer). For *l=n+1*, we define set *C* as the child nodes of root node (design element) *i*. Then, it is obvious that from the definition of relative importance indicator in (2), we have:

$$\sum_{c \in C} y_{c,i} = 1 \quad (4)$$

The total number of paths from i to its leaf nodes (ending design elements) will be the summation of the number of paths of each node $c \in C$ to their own leaf nodes (E_c) of the *direct impact-graph* for *c*. Then:

$$\sum_{e \in E} y_{e,i} = \sum_{c \in E} (y_{c,i} \sum_{e_c \in E_c} y_{e_c,c}) = \sum_{c \in E} (y_{c,i}) = 1 \quad (5)$$

■

Theorem 1: The *relative importance indicator* $y_{j,i} \le 1$ holds for any i and j in an *instance-graph*.

Proof. Without losing generality, we assume that $i \triangleright j$. We assume that j is not a leaf node. Otherwise, from Lemma 1, we have $y_{j,i} \le 1$ immediately. From Lemma 1 and (5), we know that the summation of $y(p)$ ($p \in P$) of all paths equals to 1. Here, P is the set of the paths from i to its leaf nodes. $y(p)$ is the product of the relative importance indicators of each edge of the path p. Only part of the paths will pass over node (design element) j. We denote those paths as set $P(j)$. Therefore, for the summation of $y(p)$ for $p \in P(j)$ will be less than 1, i.e. $y(P(j)) = \sum_{p \in P(j)} y(p) \le 1$.

We will prove that this summation $y(P(j))$ is $y_{j,i}$. For each path $p \in P(j)$, we split it into two parts (p_1 and p_2). The first part p_1 is from i to j, and the second part p_2 is the remaining part from j to the end of path of p. The total number of paths in $P(j)$ is the product of the number of paths from i to j, and from j to the leaf nodes (ending design elements). Let $Q(j)$ is the set of paths from i to j, then

$$\sum_{p \in P(j)} y(p) = \sum_{p \in P(j)} y(p_1) y(p_2) = \sum_{p_1 \in Q(j)} y(p_1) = y_{j,i} \quad (6)$$

The second equation comes from Lemma 1, where the summation of $y(p_2)$ for the p_2 having same p_1 equals to 1. ■

Theorem 2: The *relative importance indicator* $x_{i,j} \le 1$ holds for any nodes i and j in an *instance-graph*.

Proof. The prove process is that same as Theorem 1 ■

For a pair of $x_{i,j}$ and $x_{j,i}$ (or $y_{i,j}$ and $y_{j,i}$), only one of them could be none-zero value, which is the result of enforcing instance-graph to be acyclic.

3.3.3. Relative Importance Indicator Matrix

It is time to define a matrix called relative importance indicator matrix $D = (d_{i,j})$ as follows:

- If i and j without any path or link, $d_{i,j} = d_{j,i} = 0$
- If there is a path or link from i to j, assign $d_{i,j} = x_{i,j}$ and $d_{j,i} = y_{j,i}$
- If i and j is identical, then $d_{i,j} = d_{j,i} = 1$

Theorem 3: $d_{i,j} \le 1$ for any i and j.

Proof: It is a result from Theorems of 1 and 2. ■

Denote *I* as unit matrix, we have:

$$D = X + Y - I \quad (7)$$

3.4. Potential Engineering Implications

In this section, we explore some potential engineering usage of the relative importance indicator matrix through revisit of some well-known engineering concepts.

3.4.1. Critical Design element

The *single point of failure* [6] is a widely used concept in the IT/network communication areas and other areas to identify the constraint of certain physical layout that a system could fail at a certain point. We borrow this concept to introduce *critical design element* in SOA design. A *critical design element* is a design element that any improper design/handing of this design element will result in all other design elements be invalid. Define

mathematically, for a *critical design element*, we have the relative importance indicator $d_{a,b} = 1$ for any design element $b \neq a$ (in other word, either $b \triangleright a$ or $a \triangleright b$ has to be TRUE). From *instance-graph*, for any path from a root node (starting design element) to a leaf node (ending design element), a will be a node in the path.

With the help of above mathematical definition, we can extend the definition used for '*single point of failure*'. The following are three examples of such extension to provide the freedom for a design architects to decide what kind of design elements have higher priorities in design:

- First, we loose the requirement of $d_{a,b} = 1$ to a smaller value λ (say 0.9);
- Second, rather than ask $d_{a,b} = 1$ for any design element $b \neq a$ in the model, we can have certain percentage (say 80%) of design elements have $d_{a,b} = 1$;
- Third, we can define *critical design element bundle* (say set *A*). For any design element $b \notin A$, we have $\max_{a \in A} d_{a,b} = 1$. The intuitive interpretation is that all design elements become invalid if we remove all design elements in the bundle.

3.4.2 Model Partition

For a design of a complex system, usually it is desirable to partition the design among multiple design architects. With the help of the *meta-graph* and relative importance indicator matrix, we can achieve such partition.

Phased Sub-design - A phased design approach is to divide a design into multiple phases. It is a partition at the meta-model level. It can be achieved of re-defining *starting stereotypes* and *ending stereotypes* for each phase. Using example in Section 2 (Figure 1), we can split the end-to-end SOA design into three phases. For *service identification* phase, we re-define the *ending stereotype* as *service*. For *service specification* phase, the *starting stereotype* is *service*, and the ending stereotype is *service component*. The *starting stereotypes* and *ending stereotypes* in different phases have overlap to ensure models can be integrated smoothly back.

The *meta-graph* discussed in Section 3.1 can be used to support such partition. A *meta-graph* demonstrates the logic steps of a design architect need to follow. A reasonable split should happen at the steps which has least connection with other steps. Reflected back to the *meta-graph*, it should be the nodes with least links with nodes. From Figure 1, we know that stereotypes of *candidate service*, service and *service component* satisfy such requirement.

Design Model Partition - A more difficult partition is split a design effort vertically. It is a partition at design model level. For example, we like to have a partition based on different *business goals* (or *business domains* or *functional areas*). Such partition help design architect focusing their own portion of design task and reduce the complexity of producing an SOA solution.

A desirable partition will have as little overlap as possible for different design architects to reduce the overhead of model merge. Using a bank service as an example, a partition of the business goals of supporting "Bank Account Service" and "Small Business Loan" application is reasonable. But a partition of supporting "online transactions" and "traditional office" might not appropriate, the reason is that there might be a lot of common services at the backend to support both online transactions and traditional office.

We propose a way of quantifying degree of overlap for a split using the *join impact-graph* (discussed in Section 3.2.1). Let us say there are only two *business goals* (*f* and *g*). We can generate the two *join impact-graphs* (say *F* and *G*) for each *business goal*. Each represents its own *valid design model*. We define an overlap indicator as follows:

$$o(F,G) = \sum_{i,j \in A} \sqrt{f_{i,j} g_{i,j}} \qquad \textbf{(8)}$$

A is the set of all the nodes (design elements). We define $f_{i,j}$ (or $g_{i,j}$) as the relative importance indicator of *i* over *j* in the *join impact-graph F* (or *G*) if *i* and *j* exist in *F* (or *G*). $f_{i,j}$ (or $g_{i,j}$) is 0 if *i* or *j* does not exist in *F* (or *G*). If there is no overlap between the two models, it is obvious that $o(F,G) = 0$. The value of $o(F,G)$ not only includes direct influence from design elements and their relationship (edges) for such partition, it also includes the influences of implied impact of these design elements on other design elements. If a design element does not have large impact on other design elements, the overlap of this design element will not have much contribution to the indicator of $o(F,G)$.

3.4.3. Model Coupling Indicator

SOA has been positioned as a best way of designing a loosely coupled architecture for a software system. An engineering indicator of quantifying the degree of coupling certainly is very valuable. For a design element *i*, we define:

$$c_i = \frac{\sum_{j \in \{M/i\}} d_{i,j}}{m-1} \qquad (9)$$

M is the set of design elements in model with cardinality m. Using (9), we define the overall degree of coupling c for an SOA design model as:

$$c = \frac{\sum_{i \in M} c_i}{m} \qquad (10)$$

3.3.4. Service Exposure Level Indicator

Service exposure is an accessibility level (such as *service component*, *functional area*, *division, organization* or *external*) required for a service. For certain services, we know their accessibility levels from design functional requirements. For remaining services, we need to find out their exposure level based on their relationships with other services. In the current engineering design, we simply review the candidate list to ask the question "Is this really a candidate for exposure at certain level?" Although, in theory any service could be exposed at any level, but the cost associated with exposed service (such as governance and underlying infrastructure of the service and the components) and security constraints may not allow us to do so. We will demonstrate how the relative importance indicator matrix can help make such decision.

For a given level accessibility, say *organization*, we denote S as the set of those services identified in functional requirement to be exposure. Then, for a service (say i) not in S, we define an indicator e as the degree of the necessity of exposing the service i at same level of exposure.

$$e = \sum_{j \in S} d_{i,j} \qquad \mathbf{(11)}$$

3.4.5. Towards Variation Oriented Analysis and Design

We know that the relative importance indicator matrix is a quantitative measure of impact of a design element on other element. In variation oriented analysis design [2], many design elements could be impacted when one design element is changed. This indicator can be used to prioritize the effort of VOAD. Say if design element a is changed, those design elements in the *join impact-graph* for a with higher value of relative importance indicator require more attention with high priority. The prioritization is very important for a complex SOA solution design. In theory, the number of impacted design elements could increase exponentially as model become more complex.

4. Related Work

Software architecture modeling has become a research topic since the beginning of software research. The modeling techniques include free-text annotation, power point charts, and binary metadata. More formal methods have XML description languages [7], UML modeling [3], and some other abstracting techniques such as Petri Nets [8] and Architecture Description Language (ADL) [9]. Applications using those architecture models or languages spanned in various application areas. Petty et. al. [9] used a software ADL to model a federated system's architecture and run-time performance. Architecture-based performance analysis was applied in the telecommunication system [10]. Paper [7] also presented an XML-based architecture description language. A unification framework was proposed to perform architecture-based software reliability prediction based on analytical models [11].

Mathematical modeling approaches were also explored by the researchers in the field. A Hidden Markov Model was used to present a runtime self-adaptation method in software-intensive systems [12]. To analyze software architecture, a formal method was proposed for models in SAM [13]. Petri Nets were used to model behavioral patterns of concurrent software architectures [8].

Integrating business requirements and goals with software architecture modeling is an important research direction. Paper [14] proposed a goal-oriented design of business models and software architectures.

Some software architecture modeling platforms were created for various scenarios such as a software architecting environment in [15]. Based on Model Driven Architecture (MDA), software architecture related aspects were discussed in [16]. SoftArch was used in modeling and analyzing software architecture [17].

Service-Oriented Architecture (SOA) is becoming an important type of software architectures for supporting loosely-coupled integration for modules that could be deployed on different platforms. A modeling framework for SOA was presented in [18].

However, none of those related works have addressed the mathematical formation for the emerging SOA design space. Meanwhile, from engineering perspective, Rational Software Architect (RSA) supports displaying the relationship of the UML design element using *Browse & Topic Diagrams* to support impact analysis. This generic capability needs to be extended to support of domain-specific awareness in SOA context. For example, the notions of ending point and expansion levels are handled only in terms of the basic UML concepts, and are not easy to allow for domain-specific criteria (e.g. stereotypes) to be factored into the analysis.

5. Conclusion and Future Directions

We presented our mathematic formulation of meta-model and design model as directed graphs, deduced a

matrix to quantify the relative importance of two design elements, and discussed some potential engineering implications of using this matrix. This framework can be applied into engineering practices such as enhancement of SOMA-ME tool discussed in Section 2 and other consulting practices. This approach might be also applicable in other model-driven modeling methodology, such operational modeling, where the relationship is focusing on infrastructure aspects of an SOA solution deployment and management.

From business aspect, we only utilized the structural relationship of the design elements. The semantics of the design elements and their relationships have not been fully utilized in our analysis. This could be a direction for the extension of the proposed mathematical framework. Application into engineering practice to provide useful indicators and metrics certainly need more study.

From theoretical point of view, a potential interesting topic could be design model robustness and model reduction analysis. For example, model robustness could be the analysis of the impact of a service change to overall IT reliability to support desired business goals. Similar to geographic map zooming in and out, we can hide or visualize certain complexity of the model for a design architect by grouping certain design elements and links. An interesting topic is to quantify the complexity change due to such model reduction.

6. References

[1]. Liang-Jie Zhang, Jia Zhang, Hong Cai, Services Computing, Springer and Tsinghua University Press, July 2007.

[2]. Liang-Jie Zhang , Ali Arsanjani , Abdul Allam , Dingding Lu , Yi-Min Chee , Variation-Oriented Analysis for SOA Solution Design, *IEEE International Conference on Services Computing* (SCC 2007), July 2007, pp. 560-568.

[3]. Unified Modeling Language, http://www.uml.org, OMG.

[4]. Liang-Jie Zhang, Nianjun Zhou, Yi-Min Chee, Ahamed Jalaldeen, Karthikeyan Ponnalagu, Renuka R Sindhgatta, Ali Arsanjani, Fausto Bernardini, SOMA-ME: A Model-Driven SOA Solution Design Platform, submitted to *IBM Systems Journal*, Special Issue on Services Architectures, January 2008.

[5]. *Wikipedia* –Topological sorting http://en.wikipedia.org/wiki/Topological_sort.

[6]. *Wikipedia* - Reliability engineering http://en.wikipedia.org/wiki/Single_point_of_failure .

[7]. Eric M. Dashofy, Andre van der Hoek, Richard N. Taylor, A Highly-Extensible, XML-Based Architecture Description Language, *Proceedings of Software Architecture*, 2001, pp. 103-112.

[8]. Robert G. Pettit IV, Hassan Gomaa, Modeling Behavioral Patterns of Concurrent Software Architectures Using Petri Nets, *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture* (WICSA'04), pp. 57-66.

[9]. Mikel D. Petty, Frederic D. McKenzie, and Qingwen Xu, Using a Software Architecture Description Language to Model the Architecture and Run-Time Performance of a Federate, *Proceedings of the 6th IEEE International Workshop on Distributed Simulation and Real-Time Applications* (DS-RT'02), pp. 85-92.

[10]. Dorina Petriu, Christiane Shousha, Anant Jalnapurkar, Architecture-Based Performance Analysis Applied to a Telecommunication System, *IEEE Transactions on Software Engineering*, vol. 26, no. 11, November 2000, pp.1049-1065.

[11]. Swapna S. Gokhale,Kishor S. Trivedi, Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework, IEEE *Transactions on Reliability*, Vol. 55, No. 4, Dec. 2006, pp.578-590.

[12]. Hua Wang and Jing Ying, Toward Runtime Self-adaptation Method in Software-Intensive Systems Based on Hidden Markov Model, *COMPSAC 2007*, Vol. 2., pp. 601 – 606.

[13]. Huiqun Yu, Xudong He, Yi Deng, Lian Mo, A Formal Method for Analyzing Software Architecture Models in SAM, *Proceedings of the 26 the Annual International Computer Software and Applications Conference* (COMPSAC'02), pp. 645-652.

[14]. Deryck A. Velasquez, Deryck A. Velasquez, Goal-Oriented Design of Business Models and Software Architectures, *IEEE CCECE/CCGEI*, Ottawa, May 2006, pp.510-513.

[15]. Claudio Riva, Petri Selonen, Tarja Systä, Antti-Pekka Tuovinen, Jianli Xu, Yaojin Yang, Establishing a Software Architecting Environment, *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture* (WICSA'04), pp. 188-197.

[16]. Nourchene Elleuch, Adel Khalfallah, Samir Ben Ahmed, Software Architecture in Model Driven Architecture, *3rd International Symposium on Computational Intelligence and Intelligent Informatics* – ISCIII 2007, Agadir, Morocco, March 28-30, 2007, pp. 219-223.

[17]. John Grundy, Software Architecture Modelling, Analysis and Implementation with SoftArch, *Proceedings of the 34th Hawaii International Conference on System Sciences* ,2001, pp.1-9.

[18]. Tao Zhang, Shi Ying, Sheng Cao, Xiangyang Jia, A Modeling Framework for Service-Oriented Architecture, *Proceedings of the Sixth International Conference on Quality Software* (QSIC'06), pp. 219-226.

Journal of Grid Computing **00:** 1–20, 2005.

Requirements Driven Dynamic Services Composition for Web Services and Grid Solutions

Liang-Jie Zhang[1] and Bing Li[2]
[1]*IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA*
E-mail: zhanglj@us.ibm.com
[2]*Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287, USA*
E-mail: libing@asu.edu

Key words: business process composer, business requirements annotation, genetic algorithm (GA), Grid computing, service composition, service selection, Web services cluster, Web services outsourcing manager (WSOM) framework

Abstract

In this paper, we present the Web Services Outsourcing Manager framework via a mathematical model for dynamic business processes configuration using existing Web Services to meet customers' requirements. An XML-based annotation document is proposed to capture the business requirements and used to dynamically generate search scripts for an advanced Web services discovery engine to find Web services from both UDDI registries and Web Services Inspection Language documents. A list of available Web services is returned for further composition and optimization to produce the final business process. This paper proposes a novel mechanism to map a service selection problem into a solution space {0, 1} to utilize global optimization algorithms such as Genetic Algorithms (GA). A working research prototype has been implemented to demonstrate the feasibility of the on-demand Web services flow composition for e-business and Grid solutions.

Abbreviations: WSFL – Web Services Flow Language; BPEL4WS – Business Process Execution Language for Web Services; BPOL – Business Process Outsourcing Language; WSOM – Web Services Outsourcing Manager; UDDI – Universal Description, Discovery, and Integration; DTD – Document Type Definition; ECA – Event–Condition–Action; GUI – Graphic User Interface; BE4WS – Business Explorer for Web Services; BBR – Business–Business Relationship; BSR – Business–Service Relationship; SSR – Service–Service Relationship; BOR – Business–Operation Relationship; SOR – Service–Operation Relationship; OOR – Operation–Operation Relationship; WSRL – Web Services Relationship Language; USML – UDDI Search Markup Language; QoS – Quality of Service; GA – Genetic Algorithm; WSRF – Web Services Resource Framework; OGSA – Open Grid Services Architecture; OGSI – Open Grid Services Infrastructure.

1. Introduction

The emergence of Web services [1] paves the way for easier business process integration and management. Web services has been used for implementing business processes through the selection and binding of Web services to business process tasks. To standardize the specification of business processes, several business process languages have been created. These languages embrace the utilization of Web services, in their functional ability to define processes, partners, task execution choreography, and fault handling. Examples of such languages are XLANG and Web Services Flow Language (WSFL), which have been superseded by Business Process Execution Language for Web Services (BPEL4WS, a.k.a. BPEL) [9]. Additional languages include Web Service Choreography Interface and others jointly defined by major e-business companies [2].

The core OGSI (Open Grid Services Infrastructure) specification in Open Grid Services Architecture (OGSA) [15, 16] has been included in Web Services Resource Framework (WSRF) [17]. The current WSRF includes WS-ResourceLifetime, WS-ResourceProperties, WS-Notification, WS-RenewableReferences, WS-ServiceGroup, and WS-BaseFaults. Moreover, Grid computing is moving to the higher-level resource sharing: business services or business process sharing [18]. Successful e-business applications in Grid will be the key driving force to move Grid computing to the next step: Business Grid. The success of the next generation Grid computing, Business Grid, will be highly dependent on the maturity of the evolving e-business computing technology such as business process modeling, integration and management. Therefore the framework presented in this paper can be directly used in Grid computing environment.

An important aspect when creating or updating a business process is to meet the new and evolving business requirements. However, current Web services based business process execution languages do not adequately accommodate detailed requirement specification, making it difficult to generate optimal business process compositions. In this aspect, there are several issues. First, business requirements and preferences tend to be informal, subjective, and difficult to quantify, which poses great challenges in automating the business process composition while incorporating business requirements. Therefore, it is critical to properly formulate the descriptive and subjective requirements in quantifiable and objective machine-readable formats to enable dynamic business process composition. Second, relationships between businesses that provide services are also an important aspect to be taken into consideration when composing a business process. Third, a single Web service is most likely inadequate to serve the customers' business needs; it takes a selection of various Web services composed together to form a business process. The challenge is to select the services that not only satisfy the individual requirements but also best fit the overall composed business process. Therefore, the entire business process needs to be optimized prior to execution. In this paper, a mathematical model using Genetic Algorithms is proposed to further optimize business process based on an available list of Web services obtained by looking up UDDI registries using customers' requirements.

Again, the challenging issues of dynamically configuring a new business process using existing Web services are as follows:

- Lack of uniform representation to capture business requirements, preferences; Web services features, event-action mapping as well as the relationships among Web services.
- Lack of an automated mechanism to generate search script to dynamically discover appropriate Web services for performing a specific task from UDDI registries populated with Web services records.
- Lack of seamless integration mechanism for template based business process flow composition and event-driven business process flow composition.
- Lack of effective service selection mechanism to automatically construct an optimal business process using available Web services.
- Lack of efficient tooling to support dynamic adaptation of Web services flow to different modeling languages (e.g., BPEL, WSFL).

This paper proposes some novel mechanisms in solving the above mentioned challenging issues. First, an XML-based language Business Process Outsourcing Language (BPOL) is proposed to describe the business requirements and preferences. Next, an dynamic two-level service selection is performed to optimize the generated business process. The level-one service selection is performed through the Web services search script, which is automatically generated based on BPOL, and then invoked to search preferred UDDI registries to obtain the list of qualified Web services. The level-two service selection is an optimization process using an optimization algorithm, Genetic Algorithm, taking into account the relationships among the services.

The rest of the paper is organized as follows: First, there is described BPOL, followed by the architecture of the Web Service Outsourcing Manager, business process outsourcing model, and its major components; next, business process outsourcing model and its adaptation using the global optimization algorithm – Genetic Algorithm – is presented. Last, we briefly introduce a research prototype that automates a business process composition and optimization, followed by related work and conclusions.

2. BPOL

Business Process Outsourcing Language (BPOL) is an XML-based annotation language for business requirements representation, including service flow rules,

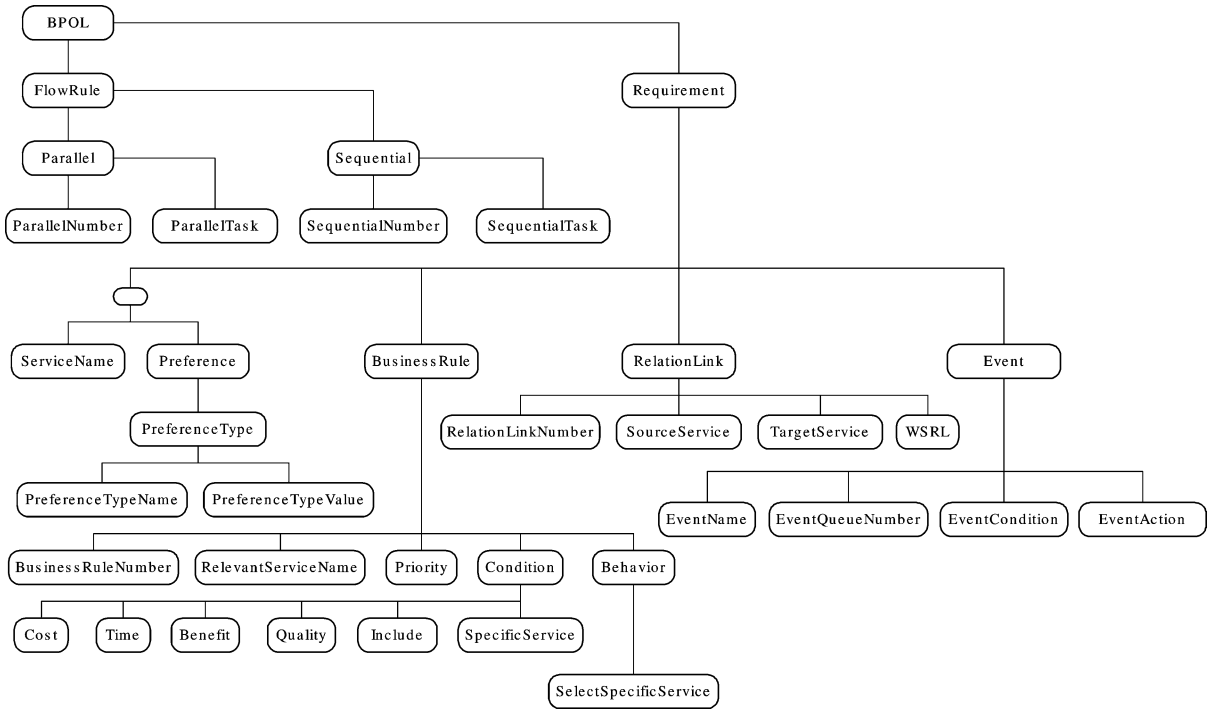


Figure 1. BPOL data structure.

customer preferences, and business rules. The business requirements and preferences specified by BPOL can be used for service discovery and selection.

As illustrated in Figure 1, the major elements of this annotation are described as follows: service name, preference, business rules binding, service relationship (RelationLink), and event (for binding as well).

In Web Service Outsourcing Manager framework, BPOL is the core input to Business Process Composer component, and it can be automatically generated by the Requirement Analyzer component, which is shown in Figure 3.

There are two major parts in BPOL, i.e., the flow rules and business requirements. Flow rules include parallel services and sequential services requested by customers. The input to Web Service Outsourcing Manager framework is the expected business processes provided by customers. The WSOM creates all the possible parallel and sequential services from those expected processes.

Next, major elements of this annotation language are described as follows: service name, preference, business rules binding, service relationship (RelationLink), and event (for binding as well) Preferences include sets of name and value pairs, such as the preferred UDDI registry name and location link to search. Business rules govern the selection of a Web service and optimal business process; they include cost, time, benefit or service bonus, quality of service, and specific or preferred services. Service relationships describe the business relationships between service providers. For example, if service A is selected together with service B, the combined cost is less than if they were not selected together, thus affecting an the overall cost in a business process composition. A Web service may include multiple operations that conform to a certain invocation-sequencing rule. A binding event associated with these operations in a Web service triggers an event action to be performed for evaluating the service selection.

Note that the policy for a virtual organization (VO) in Grid computing environment could cover all aspects of a VO [21]. The example aspects include statements of its purpose, operations, involved resources, and users [21]. The main purpose of Grid policy is to provide guidelines and control the resource sharing. However, the specific business preferences captured in BPOL are more focused on the business requirements and services binding for compositing a business process. We can think that the context of business composition is a marketplace or private exchange,

List 1. BOPL's DTD

```
<!ELEMENT BPOL (FlowRule*, Requirement*)>
<!ELEMENT FlowRule (Parallel*, Sequential*)>
<!ELEMENT Parallel (ParallelNumber, ParallelTask*)>
<!ELEMENT
ParallelNumber (#PCDATA)> <!ELEMENT ParallelTask
(#PCDATA)>
<!ELEMENT Sequential (SequentialNumber,
SequentialTask*)>
<!ELEMENT SequentialNumber (#PCDATA)>
<!ELEMENT SequentialTask (#PCDATA)> <!ELEMENT
Requirement ((ServiceName, Preference)*,
BusinessRule*, RelationLink*, Event*)>
<!ELEMENT ServiceName (#PCDATA)>
<!ELEMENT Preference (PreferenceType*)>
<!ELEMENT PreferenceType (PreferenceTypeName,
PreferenceTypeValue)>
<!ELEMENT PreferenceTypeName (#PCDATA)> <!ELEMENT
PreferenceTypeValue (#PCDATA)> <!ELEMENT BusinessRule
(BusinessRuleNumber, BusinessRuleType,
RelevantServiceName, Priority, Condition, Behavior)>
<!ELEMENT BusinessRuleNumber (#PCDATA)>
<!ELEMENT BusinessRuleType (#PCDATA)>
<!ELEMENT RelevantServiceName (#PCDATA)> <!ELEMENT
Priority (#PCDATA)>
<!ELEMENT Condition (Cost*, Time*, Benefit*, Quality*,
Include*, SpecificService*)>
<!ELEMENT Cost (#PCDATA)>
<!ELEMENT Time (#PCDATA)>
<!ELEMENT Benefit (#PCDATA)>
<!ELEMENT Quality (#PCDATA)>
<!ELEMENT Include (#PCDATA)>
<!ELEMENT SpecificService (#PCDATA)>
```

which can be treated as a VO. Therefore, the BPOL proposed in this paper could be a subset or a value-added part of the policy to be defined for a VO in Grid computing environment.

The structure of the BPOL is shown in DTD (Document Type Definition) form in List 1.

Business rules specify how the various steps of a business process can be encoded in the form of reaction rules. Note that different kinds of business rules have been extensively used in database engineering and e-business solutions. From [21] and [22], we can see that some types of business rules could be embedded into a policy. In this paper, business rules are part of the customer's requirements.

The event list defined in BPOL is used to capture the event-condition-action (ECA) mapping for event driven business process composition. In fact, the extensible structure of BPOL enables the importing of existing XML files, such as FlowXML file, BusinessRuleXML file, PreferenceXML file, and ECA-XML for business flow, business rules, preferences, event-action mappings respectively. In addition, BPOL not only acts as a container of these existing XML formats but also carries the annotation information among the objects listed in different XML files.

In addition, it is important that vocabularies for business requirements are clearly and consistently defined among services, and registries as well as used by customers. Two of the possible approaches for obtaining business requirements are: (1) through user input from the Graphic User interfaces (GUIs), and (2) converting other requirement documents into BPOL. When using a GUI to collect requirements, users are presented with a set of pre-defined vocabulary terms to select from. Therefore, the consistency in terminology is easier to achieve. However, when converting other requirements to BPOL, different vocabulary may be used to mean different meanings; ambiguity tends to arise, and the correct meaning can get lost during the conversion. For example, a variation of the meaning of "Cost" may be "Price", "Charge", or "Fee", etc.

It is necessary for BPOL files to be exchanged among different systems. Thus, having a unique and standard vocabulary library is critical to avoid misunderstanding in describing business requirements. A standard vocabulary library facilitates systems to collect business requirements, search useful services from the registries, and obtain the optimum business processes for customers. But in fact, it is difficult to constrain every customer or every service provider to select uniform terminology to define their requirements or services. Two solutions are put forward: (1) building a predefined vocabulary library from on a list of frequently used terms based on available services and requirement descriptions, and (2) adding new terms to the vocabulary. In the predefined vocabulary library, participants in a business process can select the correct term to represent their meanings. In the second solution, when a selecting from predefined terms via GUI and the predefined terms like "Cost" is deemed insufficient, a new term can be added via the GUI. For example, a new term a new term "Cost-PlusTax" can be added if none is available. However, rules must be conformed when defining new terms so that they can be processed correctly. There will be checks to detect duplication and non-conformance of the creation rules. Note that the concept of vocabulary library is similar to ontology, which is widely used in semantic Web community.

Synonyms can be incorporated into the library in cases where customers and providers prefer to use their own instead of the existing vocabularies with

same meaning in the library. Once new vocabularies are added into the vocabulary library, they become standards for future use. Currently, some examples of the predefined vocabularies in the library are as follow:

- *Cost* represents how much customers need to pay for their businesses;
- *Time* – how long it will take customers to finish their businesses;
- *Bonus* – how much would the cost be reduced if customers select a particular service;
- *Quality* – quality of service;
- *Binding* – additional services that might be provided if a particular service is selected by customers.

Next, two components used by Web Service Outsourcing Manager framework are introduced, namely, Web Services Relationship Language WSRL [5], which is an example of the regular XML-based annotation information and Business Explorer for Web services [3].

Further, the basic information of Web services is described in WSDL. An important part of the data about Web services is the relationships among business entities, business services and operations, which are keys to composing and executing dynamic business processes. However, the current Web services specifications and UDDI specification lack in the definitions and descriptions of such relationships. WSRL, as described in [5], captures the Web services relationships at different granularities: business–business relationship (BBR), business–service relationship (BSR), service–service relationship (SSR), business–operation relationship (BOR), service–operation relationship (SOR), operation–operation relationship (OOR). These relationships such as partnership and alliances facilitate selecting and composing a set of services that meets the business requirements. In the BPOL specification, the WSRL is the input of the BPOL composer. The WSRL can be embedded into BPOL via a link tag.

An XML based UDDI exploration engine, Business Explorer for Web services (BE4WS) used in the WSOM, provides developers with standard interfaces for efficiently searching business and service information in one or more UDDI registries, and it is part of the IBM Emerging Technologies Toolkit [4]. The engine is based on the proposed XML-based search script, UDDI Search Markup Language (USML), to represent a search request including multiple queries, key words, UDDI sources, and aggregation operators. BE4WS processes the USML request and performs advanced UDDI registry exploration and aggregates search results from different UDDI registries.

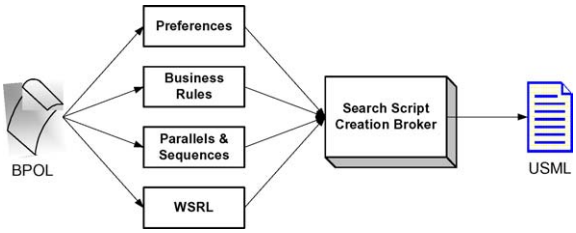


Figure 2. USML generation diagram.

List 2. Sample BPOL

```
<?xml version = "1.0"?>
<!DOCTYPE BPOL SYSTEM "BPOL.dtd" >
<BPOL>
<FlowRule>
<Parallel>
<ParallelNumber>0</ParallelNumber>
<ParallelTask>Match</ParallelTask>
<ParallelTask>Food</ParallelTask>
</Parallel>
.....
<Sequential>
<SequentialNumber>5</SequentialNumber>
<SequentialTask>Start</SequentialTask>
<SequentialTask>News Collection</SequentialTask>
<SequentialTask>Food</SequentialTask>
<SequentialTask>End</SequentialTask>
</Sequential>
.....
</FlowRule>
.....
</BPOL>
```

Business requirements described in BPOL are important information in creating USML because multiple search queries come from business needs and preferences. The relationships between search queries can be also obtained from the Parallels and Sequences among Web Service clusters. Business rules in BPOL can help the broker generate more precise USML to retrieve qualified Web services.

In Figure 2, a USML Generation Diagram, there is shown how a Search Script Creation Broker in BPOL Processor is developed to automatically construct a USML script based on BPOL for forwarding to the advanced Web services discovery engine (BE4WS).

List 2 shows an example of a BPOL, which gets translated into USML in List 3.

Based on the flow rules in the BPOL, categories derived from the task names, in conjunction with preference files, are used for generating search scripts. For example in List 3, a task of “Food” with preference of “sit-down dinner” will map into the NAICS category of “Full-Service Restaurants” or 72211; a task of “Match” will map to the NAICS category of

List 3. Sample USML scripts generated based on the BPOL in List 2

```
<?xml version = "1.0"?>
<!DOCTYPE Search SYSTEM "UDDISearch.dtd">
<Search>
<ProcessID>0001</ProcessID>
<Query>
<Source>Public UDDI</Source>
<SourceURL>http://wsbi05/services/uddi/inquiryAPI
</SourceURL>
<BusinessName>%</BusinessName>
<Category type = "NAICS">72211</Category>
<FindBy>Business</FindBy>
</Query>

<Query>
<Source>Private UDDI</Source>
<SourceURL>http://wsbi10/services/uddi/inquiryAPI
</SourceURL>
<BusinessName>%</BusinessName>
<Category type = "NAICS">71121</Category>
<FindBy>Business</FindBy>
</Query>
<Query>
<Source>Public UDDI</Source>
<SourceURL>http://wsbi04/services/uddi/inquiryAPI
</SourceURL>
<BusinessName>%</BusinessName>
<Category type = "NAICS">5133</Category>
<FindBy>Business</FindBy>
</Query>
<AggOperation>OR</AggOperation>
</Search>
```

"Spectator Sports" or 71121; a task of News Collection will map to the NAICS category of "telecommunications" or 5133. In addition, the preference file also indicates which UDDI registry locations to search for the Web services, i.e., private eMarket A located at a particular URL (e.g., http://wsbi10/services/uddi/inquiryAPI,) or public UDDI registry located at another URL (e.g., http://wsbi05/services/uddi/inquiryAPI.). USML search script as listed in List 3, below, is generated based on the requirements in List 2.

This USML script will aggregate results via aggregation operator OR from all three searches, with each denoted with the pair of <Search> and </Search> tags, and return the results to caller at the same time.

3. WSOM Architecture

Grid computing provides an infrastructure that enables resources to be shared and accessed in distributed environments. The resources include computing resources (e.g., CPU), storage resources (e.g., hard disk), as well as Grid services. Grid service was a core concept of OGSA. With the introduction of WSRF, all the high-level resources, Grid services, have been replaced by Web services [21]. General speaking, the WSOM framework can be directly applied to any service oriented Grid computing environments by customizing the business requirements. In the rest of the paper, we will focus on the resource aggregation for compositing a business process in a pure Web services scenario. The other scenarios such as computing power and storage sharing issues will not be further discussed in this paper.

The architecture of the Web Service Outsourcing Manager (WSOM) framework is shown in Figure 3. The WSOM is a two-level service selection mechanism for narrowing down the desired services. The first level is achieved by using the advanced Web services discovery mechanism using business requirements specified in BPOL as input and automatically generate XML-based search script to find Web services. The second level is achieved by using both business requirements and the optimization algorithm to select the best suitable Web services from the initial list created by the first level and optimize the entire business process based on the business requirements. The intermediate output of WSOM is BPOL with the one of the potential final output being Web Services Flow Language (WSFL) or BPEL, which describe the process execution of Web services.

The detailed steps involved may be described as follows with reference to Figure 3:

Step 0. Web services are published to a centralized public UDDI registry or multiple private UDDI registries for discovery.

Step 1. Customer sends out business requirements from the application to the Requirement Analyzer.

Step 2. Requirement Analyzer creates BPOL based on the requirements.

Step 3. BPOL processor parses BPOL and automatically generates XML-based search script (e.g., USML) for the advanced Web services discovery engine.

Step 4. The Advanced discovery engine conducts search process across multiple UDDI registries or WS-Inspection documents [27] based on the search criteria specified in the USML script created by BPOL Processor. The aggregated result is a list of available services that meet the search criteria.

Step 5. The available service list is passed to Business Process Composer for service composition.

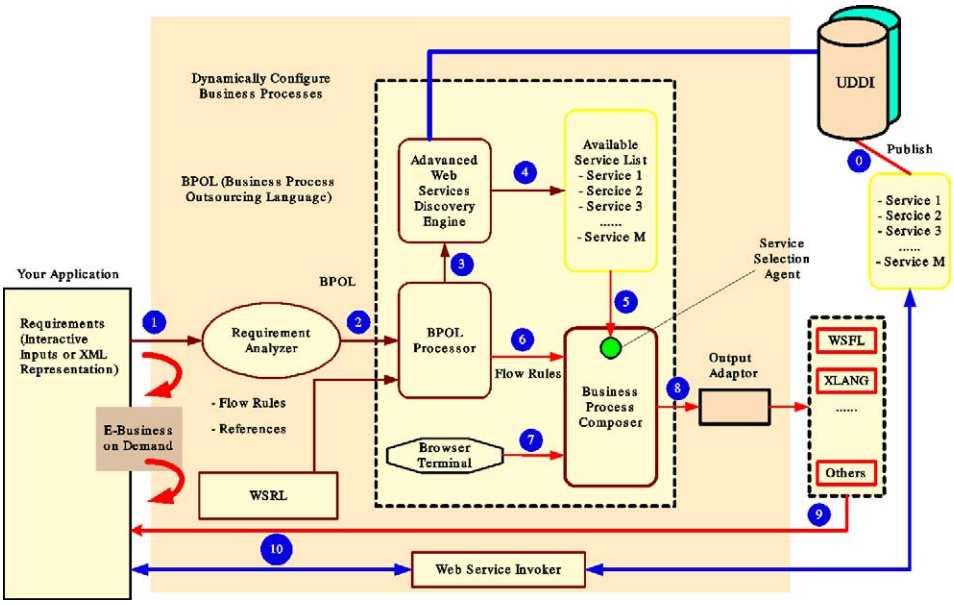


Figure 3. Business process outsourcing model.

Step 6. BPOL processor uses WSRL as input to extract the business flow rules from BPOL and pass the result to Business Process Composer.

Step 7. Decision makers can interact with Business Process Composer via Web Browser or GUI to tune the service selection and composition process.

Step 8. The result of Business Process Composer is formatted as a target Web service execution language such as BPEL or WSFL defined for Web services via Output Adaptor.

Step 9. The resulted Web service execution language from Output Adaptor is returned to the application.

Step 10. When the application wants to access and uses this newly composed Business Process, a business process execution engine will dynamically invoke the respective Web services, which are part of the newly created business process.

4. Business Process Outsourcing Model

The goal of business process outsourcing is to configure a real-life business process using the dynamic services discovery, selection, and optimization mechanisms to meet business requirements.

A Web services cluster is a conceptual Web Service, which is not referred to a real or concrete Web Service. A cluster represents a collection of available Web services provided by multiple service providers to perform a specific function. For example, shopping Web Service, shipping Web Service and credit checking Web services are different Web services clusters. At the end of the service selection and business process composition process, each Web services cluster will be mapped to a concrete or real Web Service.

A dynamic business process composition showing Web services clusters is illustrated in Figure 4.

This is a typical business process flow comprising multiple Web services clusters as well as a business start and a business end. For each Web services cluster, there are multiple candidate services provided by multiple service providers.

Next, we'd like to introduce the concept of a "service set", which is referred as a collection of Web services in this paper. A typical service set is shown as follows:

$$S_i = \{s_1, s_2, \ldots, s_{j(i)}\}, \quad 1 \leqslant j(i) \leqslant N, \qquad (1)$$

where S_i is represented as a service set, Service 1 as s_1, Service 2 as s_2, and Service N as s_N, etc. The services specified in one service set are to be configured to construct a new business process. Note that the sequence of the services and the relationships among services in the business process flow is defined in BPOL. Note that it is usual that two identical services can be defined in the same service set. In other words, two

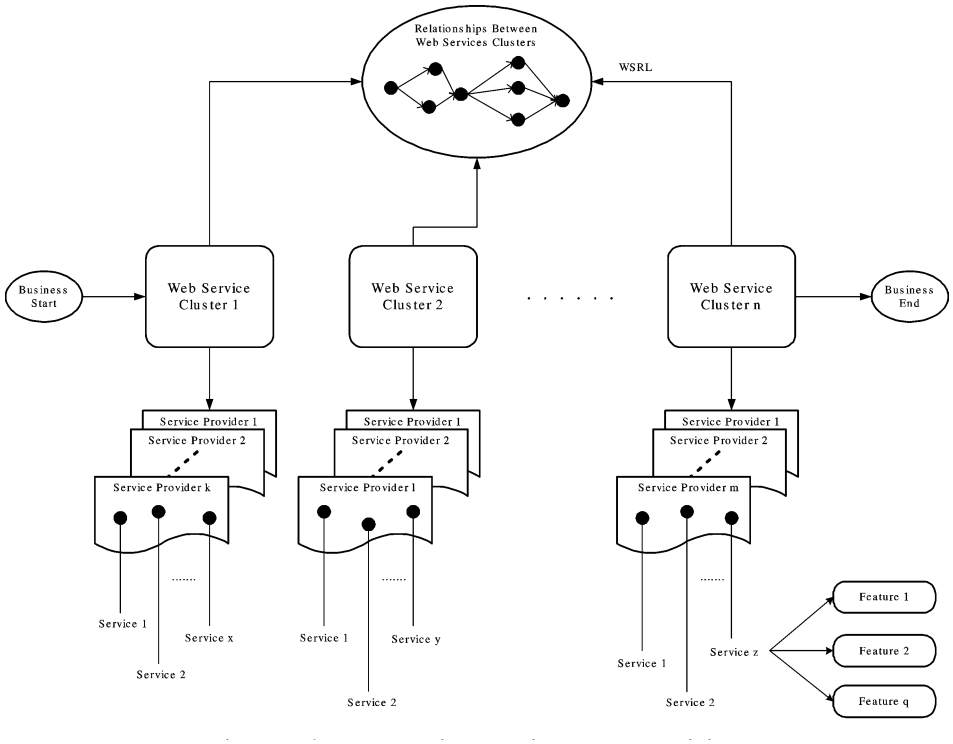


Figure 4. Dynamic service composition.

identical services can be used at different stages of a newly created business process. So it is recommended that all the identical services used at different stages should be put into one service set.

Theoretically, each business process can be constructed at minimum by one service. A typical business process comprising a set of services, service set, is represented as follows.

$$BP_i = h(S_i), \quad 1 \leqslant i \leqslant M. \tag{2}$$

Here, h represents the complicated construction schema defined in BPOL for configuring a business process based on service set S_i. If there are only N Web services in the service set, then the total number of the possible business processes M can be defined as the factorial of N, namely, for example, if there are *six* Web services ($N = 6$), a new business process can be constructed with 720 potential combinations ($M = 6! = 720$). In our model, the relationship and data connectivity between two sequential services are used to construct a new business process. The above representation just means that the business process is finally constructed by $j(i)$ services. The number of $j(i)$ shown in (1) depends on the business requirements such as preferences, flow rules and other requirements defined in BPOL.

For a specific service, e.g., s_p, only one choice will be made by the WSOM: either selected or deselected. Theoretically, each business process can be constructed by one service, two services or N services, shown as follows:

$$BP_i = h(\{s_1, s_2, \ldots, s_p\}), \quad 1 \leqslant p \leqslant N,\ 1 \leqslant i \leqslant M, \tag{3}$$

where $s_p = \{0, 1\}$, $p = j(i)$. For example, the ith candidate business process is constructed by the following service set:

$$S_i = \{0, 1, 0, 1, 1\}, \qquad N = 5, \tag{4}$$

where '1' denotes that the service is selected. That means, only service 2, service 4 and service 5 have been selected by the WSOM. For each business process, the expected result R_i^* is described as follows:

$$R_i^* = g(BP_i) = gg(S_i). \tag{5}$$

A functional relation $g(BP_i)$ or $gg(S_i)$ returns a measure of quality, or fitness, associated with the business process composition model. The constructed business processes satisfying business requirements are shown as follows:

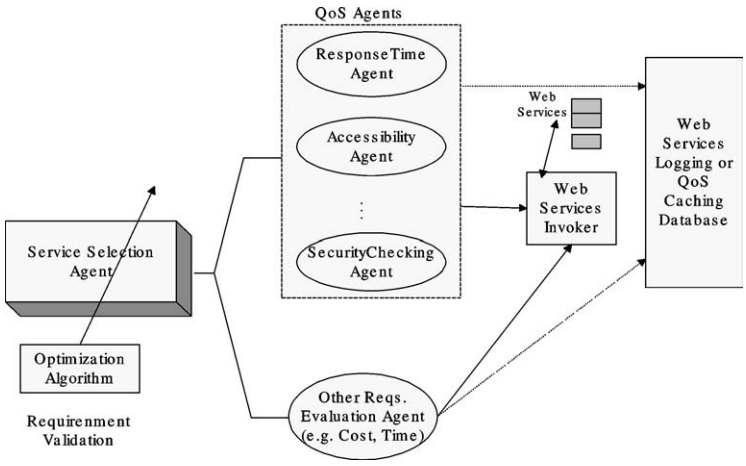


Figure 5. The diagram of Service Selection Agent.

$$BP_{\text{matched}} = Best\{BP_1, BP_2, \ldots, BP_M\}, \qquad (6)$$

where BP_{matched} denotes the best business processes, comprising some of the matched services set S^*.

So the service selection procedure is equivalent to picking up the appropriate services from the available service list which could be dynamically created by the advanced Web services discovery engine. Therefore, the service selection criterion is a key issue when configuring a new business process. In this paper, a sample optimal business process construction criterion is defined as the one that most closely matches business requirements as measured by the total error function f, described as follows:

$$f(S) = \sum_{i=1}^{i=P} E_i = \sum_{i=1}^{P} \left[\frac{1}{2} (R_i^{\text{d}} - R_i^*)^2 \right]. \qquad (7)$$

This cost function includes multiple variables. The number of the variables is determined by the business process flow template or the dynamic behaviors of the event-driven business process. So the following equation represents minimizing the least squares fitting problem.

$$\begin{aligned} \min\{f(S)\} &= \min\left\{ \sum_{i=1}^{i=P} E_i \right\} \\ &= \min\left\{ \sum_{i=1}^{P} \left[\frac{1}{2} w_i (R_i^{\text{d}} - R_i^*)^2 \right] \right\}. \end{aligned} \qquad (8)$$

Here $\forall S \in \{0, 1\}^N, 0 < f(S) < \infty, 0 \leqslant w_i \leqslant 1$ and $f(S) \neq const$, where P is the number of customer requirement indicators, w_i represents the weight of the ith customer requirement indicator, R_i^{d} is the target requirement indicator of the ith requirement and R_i^* represents the estimated value derived from the WSOM. The requirements are defined in BPOL. Note that both R_i^{d} and R_i^* are normalized for evaluating the quality of the constructed business process. To normalize R_i, it is necessary to find a standard to evaluate each R_i^{d}. In general, we can find the largest $R_{\max}$ from a particular Web Services cluster and the particular R_i. Then, set $R_i^{\text{d}} = R_i / R_{\max}$, i.e., the value of $R_i / R_{\max}$ is the normalized requirement R_i^{d}. Another approach is to select customers' expected R_{e} as the standard to normalize R_i. If R_i^{d} is more than 1, it means that the service is not the appropriate one, then we enforce $R_i^{\text{d}} = 1$. If less than 1, the service is a possible candidate to fulfill customers' requirements.

The requirements validation is an important aspect of the outsourcing, which comprises one or multiple indicators such as QoS (Quality), Cost, execution Time, and others. Typically, the execution time and cost are considered to be the quality indicators of a constructed business process. Other parameters such as availability, accessibility, etc. are also used in BPOL to specify a metric of the quality of the constructed business process.

Figure 5 describes an example procedure of checking Web service capability to validate whether the service meets the requirements.

A tool such as the Service Selection Agent employing the optimization algorithm communicates with Quality of Service (QoS) Agents or other requirement evaluation agents to perform requirement validation. The QoS Agents can use Web Services Invoker, a proxy program that invokes Web services for

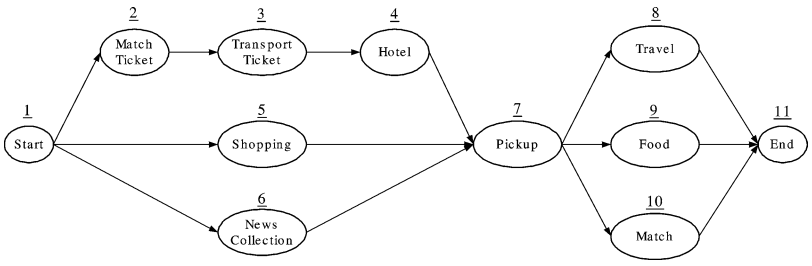


Figure 6. A basketball fan's business requirement.

a client requester (or application), to automatically invoke Web services to check the capability of that Web service or fetch data from Web Service logging or caching database to get the estimated quality.

The QoS parameters of Web services are used to measure the quality of the Web services clusters for business process composition. Typically, the QoS parameters of a Web Service are: accessibility (measured by Accessibility Agent), response time (measured by Response Time Agent), security (measured by Security Checking Agent), availability, and so forth. For example, the service selection agent tool, supported by an optimization algorithm, can get the response time by recording the invocation request time and result return time. In the meantime, the accessibility can be measured by sending an invocation request several times during a specified period, for example 24 hours.

5. GA Adapted for Service Selection

The purpose of WSOM is to automate the end-to-end composition of business processes using existing Web services. That is, using BPOL to collect business requirements, WSOM bridges the gap between the informal, subjective business requirements and the objective machine-readable business process execution language, such as BPEL4WS, WSFL, and XLANG. WSOM automatically generates search scripts for Web services lookup; and it automates the process of selecting Web services. In addition, WSOM uses a "pluggable" optimization framework to plug-in any suitable optimization algorithm to perform process optimization. In this paper, we use Genetic Algorithms as an example for service selection. Genetic Algorithms (GAs) include steps called genetic operators that are very similar to biological evolution, such as selection, reproduction, crossover, and mutation [5]. By the operator of mutation, GAs may reach a point in the solution space with non-zero probability, and they may converge to the global optimum if the best solution of a generation is always maintained in the offspring.

The optimization of service selection usually consists of finding the "point" in parameter space corresponding to the model that maximizes the fitness function. Genetic Algorithms are global and adaptive optimization techniques based on the mechanics of natural selection and natural genetics. They are often used to tackle the following optimization problems of the type:

$$\min\{f(C) \mid C \in IB^N\}. \tag{9}$$

Assuming that for

$$\forall C \in \{0,1\}^N, \quad 0 < f(C) < \infty \text{ and } f(C) \neq const.$$

Equation (9) is equivalent to Equation (8) because they have the same constraints and the same parameter space. Therefore, the second level of the service selection problem of the Web services outsourcing model can be solved by the genetic algorithm.

A chromosome represents a potential business process, which in turn represents a solution to the business requirements. Each gene in a chromosome is a service candidate with two possible values of 0 and 1. If the value of gene is 1, it means the service is selected. If 0, the service is not selected. The combination of those genes forms a chromosome, which is a series of selected and not selected services. Some modification of the vanilla GA took place in the WSOM-adapted GA. According to the architecture shown in Figure 4, each service a customer requires may have more than one provider and each provider has more than one service. As may be explained later with reference to Figure 6, in the basketball fan example there are four available Match Ticket service providers (TempTicket.com, AZCompany.com, StatePress.com and Free.com), three news collection providers (tempNews.com, myNews.com and Daily.com), etc.

Therefore, when designing the chromosome, binary strings are expressed in the formula as follow

to denote the combination of different sets of Web service clusters.

$$Chromosome = [S_{11}S_{12}\ldots S_{1i} \mid S_{21}S_{22}\ldots S_{2j} \mid \ldots \mid S_{n1}S_{n2}\ldots S_{nk}]. \quad (10)$$

Here, n is the number of Web Services clusters, and i, j and k represent the number of potential services provided by one or multiple service providers for a specific Web Services cluster. S_{nk} represents a specific Web Service. Therefore $S_{nk} = \{0, 1\}$. It should be noted that at most only one service can be selected from each Web services cluster.

To get an optimum business process based on the business requirements, several issues need to be considered. First, determine the Web service clusters required to fulfill the business processes to be generated, e.g., loan application, loan validation, credit validation, etc. Second, each service cluster can potentially be represented by services from more than one provider. Third, each provider's service has its own features. Fourth, all the required services have certain relationships with respect to other services. An optimum business process solution should take into account all of the above considerations. Then two properties need to be defined, i.e., Fitness and Weights. Fitness is a value assigned to an individual that reflects how well the individual solves the task at hand. A fitness function is used to map a chromosome to a fitness value. Weight is a value assigned to a particular gene that is used to represent the importance of the gene in a chromosome. Therefore, designing the fitness function is essential to the successful use of GA. To obtain an appropriate Fitness, Weights are also important. Weights are designed based on customers' preferences and common sense in practical business procedures.

Typically, Cost and Time are two primary factors that customers are concerned about. Using a binary string or chromosome to represent a business process composition, the best one is the string that leads to the lowest cost if the cost is the primary concern. When time is the primary concern, to obtain the optimum or usually the shortest amount of time, all the sequences in the services diagram are taken into account. First, the system uses GA to determine the shortest time consumption for each chromosome sequence. Second, from all of the shortest time consumptions, the sequence with the longest time consumption is selected. This value is the shortest Time consumption for the business. The chromosome sequences are generated automatically according to business requirements. For example, after all the information necessary to generate the diagram of the Basketball fan is entered, all the sequences in the diagram are created. In obtaining the shortest Time, the sequences are compared to get the shortest Time for the entire business process.

In the GA adapted for this paper, each and every member of each generation is to be assigned to a particular GA operation. Therefore, the sum of the probabilities of all three GA operations (mutation, crossover and selection probabilities) should be 1. This ensures all the members of the population are represented in the next generation.

According to experiments, even if worse chromosomes are dropped, there is still a possibility that no best chromosomes are available. This is because the Crossover and Reproduction operations cannot bring enough new chromosomes to the next generation. If this occurs, it would be difficult to obtain an optimum result. In this situation, the Mutation operation is more important to the GA [8].

In the vanilla GA, both the Repeat Times of Best Values and the Total Generation are terminating flags. In the GA adapted for WSOM, when the fitness is kept the same for more than the Repeat Times of Best Values, it is considered the optimum, and the GA is terminated, and the Total Generation is usually set up as an unlimited value.

In addition, in the adapted GA, a variant crossover point selection mechanism, constrained crossover point selection, is used to make sure the multiple genes for a Web Service cluster can be grouped together. As individual Web services clusters, separated by "|" in (10), the potential crossover point can only be chosen in the position of separation sign "|".

As an example, the operation of the Crossover and Mutation operations may be understood with reference to Figure 7.

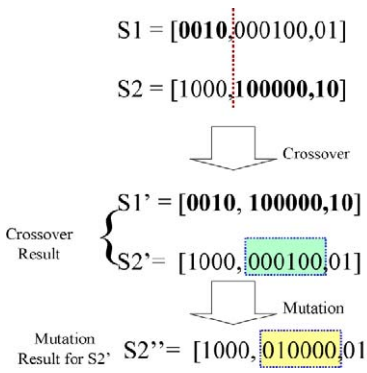


Figure 7. Example Crossover and Mutation Operations.

The chromosomes in Figure 7 have been simplified to consider only the first three Web services clusters for application of the paper to a basketball fan's travel arrangements as shown in Figures 7 and 12 and in Table 1. As shown in Table 1, the Match Ticket cluster (Web services cluster WS1 in Table 1) has four candidate services, and these services are represented by the first four binary digits in the chromosomes in Figure 7.

It will be noted that only one of these four candidate services can be selected, and therefore only one of the four binary digits has a value of "1". Similarly, the Transport Ticket cluster (Web services cluster WS2 in Table 1) has six candidate services represented by the next six binary digits in the chromosomes in Figure 7. The third Web services cluster WS3 in Table 1 is the Hotel cluster, which has two candidate services represented by the final two binary digits in the chromosomes in Figure 7. For a potential service selection, if we choose StatePress.com in the first Web services cluster (WS1) and choose US West and LA Hotel from WS2 and WS3 respectively, this service selection approach can be represented in the following gene string (a chromosome), which includes three sets of binary numbers, one for each of the three Web services clusters represented by the chromosome.

$$S1 = [0010, 000100, 01]. \quad (11)$$

Note that in the real implementation, there is no coma sign (",") between two digits. But the points in the gene string with a comma indicate the end of one cluster and the beginning of another and are potential crossover points for the crossover operation.

Suppose another potential solution is to select TempTicket.com from WS1, Regular Train from WS2, and Orange County Inn from WS3. Then we can represent this service selection solution in the following chromosome, which has the following three sets of binary numbers.

$$S2 = [1000, 100000, 10]. \quad (12)$$

When Genetic Algorithm (GA) performs crossover operation, the crossover points will be randomly selected based on a given crossover rate (e.g., a probability). For example, if the first point with coma sign has been selected for performing crossover between S1 and S2, the two chromosomes are divided at a cluster boundary and exchange their chromosome contents beyond the crossover point. Chromosome S1 is comprised of gene string before the crossover point and gene string after crossover point. Similarly, chromosome S2 is comprised of gene strings separated by crossover point. After the crossover operation the respective chromosomes S1′ and S2′ are now comprised of new gene strings, respectively. There is also a possibility for the Genetic Algorithm (GA) to perform a mutation operation for selected chromosomes (e.g., potential Web service selection solutions). The frequency for the mutation operation is defined by a mutation rate. Suppose the second binary set in S2′ (cluster WS2) will perform a mutation operation. As shown in Figure 7, chromosome S2′ shows selection of the fourth candidate service (US West) in the second cluster. After the mutation operation, the second candidate service is now shown as selected, as indicated in the mutated second cluster of chromosome S2″.

6. Research Prototype

The research prototype "Web Services Outsourcing Manager" includes two types of WSOM applications, i.e., Web Edition and Eclipse [23] Edition, providing different user interfaces but using the same WSOM engine. WSOM Web Edition is a simple Web portal for business process composition using existing Web services. WSOM Eclipse Edition provides a Java application running on the Eclipse platform for capturing requirements and configuring an optimal business process that matches customers' requirements.

A practical business process composition example is shown in Figure 6. Through this example it is easy to understand how a real life business process is configured by the paper. If a travel agency is to create a business process to fulfill all services needed so that a basketball fan from Phoenix can visit Los Angeles to watch basketball games. All services for the fan's trip as listed below need to be organized in an appropriate order, as shown in Figure 6.

- Order match tickets (Match Ticket);
- Order transportation tickets (Transport Ticket);
- Research hotels (Hotel);
- Do some shopping for the trip (Shopping);
- Collect news for the games (News Collection);
- Reserve pick-up services (Pickup);
- Make a travel schedule besides watching soccer games (Travel);
- Reserve food to enjoy the days in LA (Food);
- Reserve match services (Match).

Certain sequences exist between and among the above services, i.e., it is appropriate to perform certain services before or after others, while other services may be done in parallel. A parallel sequence means that

Table 1. A basketball fan's traveling process.

Web services cluster	Description	Candidate services	Cost (US $)	Time (day)	Normalized Cost	Normalized Time	Weighted Cost (0.6)	Weighted Time (0.4)	Service value [formula (8)]	Maximum Cost	Maximum Time
WS1 (match ticket)	Basketball fans can buy tickets from those providers	TempTicket.com	300	3	0.80	0.30	0.192	0.018	0.210	500	10
		AZCentral.com	500	2	1.00	0.20	0.300	0.008	0.308		
		StatePress.com	330	5	0.66	0.50	0.131	0.050	0.181		
		Free.com	0	10	0.00	1.00	0.00	0.200	0.200		
WS2 (transport ticket)	Basketball fans can select transportation manners from those providers	Regular train	200	2	0.31	1.00	0.041	0.200	0.241	650	2
		Express train	300	2	0.46	1.00	0.063	0.200	0.263		
		Special train	500	1	0.77	0.50	0.178	0.050	0.228		
		US West	600	1	0.92	0.50	0.254	0.050	0.304		
		Air Northern	550	1	0.85	0.50	0.217	0.050	0.267		
		US Union	650	1	1.00	0.50	0.300	0.050	0.350		
WS3 (hotel)	Two hotels are available for basketball fans	LA hotel	400	2	1.00	0.67	0.300	0.090	0.390	400	3
		Orange County Inn	300	3	0.75	1.00	0.169	0.200	0.369		
WS4 (shopping)	Three stores can be selected to order merchandises for basketball fans' trip	Tempe Center Mall	350	5	0.70	1.00	0.147	0.200	0.347	500	5
		Scottsdale City Mall	250	3	0.50	0.80	0.075	0.128	0.203		
		Arizona Mall	500	2	1.00	0.40	0.300	0.032	0.332		
WS5 (news collection)	Basketball fans need to collect news for their favorite matches	CNN.com	100	4	1.00	1.00	0.300	0.200	0.500	100	4
		ESPN.com	80	3	0.80	0.75	0.192	0.113	0.305		
		Yahoo.com	30	2	0.30	0.50	0.027	0.050	0.077		
WS6 (pickup)	Three pickup services can be chosen	Supper Shuttle	30	4	0.80	1.00	0.192	0.200	0.392	50	4
		LA Taxi	50	3	1.00	0.75	0.300	0.113	0.413		
		LA Bus	5	2	0.10	0.50	0.003	0.050	0.053		
WS7 (travel)	Two travel agencies can become basketball fans' candidates	AZ Travel Agency	1000	5	0.83	0.83	0.207	0.138	0.345	1200	6
		LA Travel Agency	1200	6	1.00	1.00	0.300	0.200	0.500		
WS8 (food)	Basketball fans need to enjoy food during the period of watching matches	Chinese food	300	2	0.50	0.50	0.075	0.050	0.125	600	4
		US food	400	4	0.67	1.00	0.135	0.200	0.335		
		Japanese food	600	4	1.00	1.00	0.300	0.200	0.500		
WS9 (match)	Basketball fans can get match services from the three providers	CA Basketball Association	500	2	0.91	1.00	0.248	0.200	0.448	550	2
		LA Student Association	300	2	0.55	1.00	0.091	0.200	0.291		
		LA Basketball Service	550	1	1.00	0.50	0.300	0.050	0.350		

services in the same set can be processed concurrently, without dependence on other services in the set. A sequential sequence means that services in the same set must be processed in a certain order. If one service is not finished, another one cannot be started.

According to the above descriptions, the Basketball fan's business requirements can be represented as a Web services cluster flow shown in Figure 6. There are nine Web services cluster to be selected for the business process composition for the basketball fan. Some services having a parallel sequence as shown in the diagram are [Match Ticket, Shopping, News Collection], [Hotel, Shopping, News Collection], and [Travel, Food, Match]. Some sequential sequences shown in the diagram are [Start, Shopping, Pickup, Food, End], [Start, News Collection, Pickup, Travel, End] and [Start, Shopping, Pickup, Match, End]. Furthermore, referred UDDI registries as specified in a preference file are searched for available services in each cluster, e.g., for the Match Ticket: TempTicket.com, AZCompany.com, StatePress.com and Free.com; for the Transportation Tickets: Train.com and Airplane.com; News Collections: CNN.com, ESPN.com and Yahoo.com, etc. Each provider also provides different types of services. For example, in TempTicket.com, customers can buy various types of tickets, the premium, the regular, and the discount tickets. Based on the business requirement, e.g., advanced Web services discovery engine, the outsourcing manager can match and select Web services, and then compose and assemble an optimal business process satisfying the business requirements.

Another criteria that can impact how service are selected to produce optimal business process is the business relationships among the service providers. As mentioned above in Section 2 (BPOL), WSRL is an XML-based description of relationships between Web services at various levels, i.e., business entity level, Web Service Level and operation level. When composing a business process, WSOM only looks at relationships at the business entity level and Web Service level, for example. There are several reasons that relationship between business entities are important in composing business processes with Web services: (1) service providers or business entities may have formed an alliance among themselves and would prefer to work with services providers within the alliance, i.e., customers may get discounts or better service; (2) business entities may be competitors and they will not work with their competitors. List 4 shows a partner

List 4. The example of WSRL between business entities

```
<?xml version = "1.0"?>
<!DOCTYPE wsrl SYSTEM "wsrl.dtd">
<wsrl>
<bbr>
<partnership id = p1>
<source>
<name> AZCompany.com </name>
<link type = "uddi-location">
<location>http://www.azcompany.com/uddi?
4C9DADD0-5C39-11D5-9FCF-BB3200333F79
</location>
</link>
</source>
<target>
<name> AZTravelAgency </name>
<link type = "uddi-location">
<location>http://www.aztravelagency.com/uddi?
4C9DADD0-5C39-11D5-3FCF-BB4500333F88
</location>
</link>
</target>
</partnership>
</bbr>
<wsrl>
```

relationship between AZCompany.com and AZTravelAgency and List 5 shows a competitor relationship between SupperShuttle and LATravelAgency.

Relationships between businesses can also be reflected at the Web Service level. At this level, certain services among service providers may be related to one another. When customers make their selections, those relationships may influence the cost, time, and the like. For example, referring again to Table 1, the Basketball fan might select Scottsdale City Mall to do shopping and reserves Chinese food because both of the two service providers cost the Basketball fan the least in shopping clusters (WS4) and food clusters (WS8), respectively. The cost is \$550(\$250 + \$300). But, if there is an alliance between Tempe Center Mall and US.

Food, the Basketball fan's selection might be changed. For example, the alliance claims that there is 30% discount if customers select both of them. The cost will be $\$525 = (\$350 + \$400) \cdot (1 - 30\%)$. Under this situation, the Basketball fan must select the alliance to fulfill his minimum cost requirement. List 6 shows the Web Service level relationships between Tempe Center Mall and US Food.

The WSOM uses the data described by the "RelationLink" element of BPOL to obtain relationship information between service providers and services and to perform relationship checking when selecting

List 5. The example of WSRL between business entities

```
<?xml version = "1.0"?>
<!DOCTYPE wsrl SYSTEM "wsrl.dtd">
<wsrl>
<bbr>
<exclusion id = ex1>
<source>
<name>SuperShuttle</name>
<link type = "uddi-location">
<location>http://www.supershuttle.com/uddi?
4C9DADD0-5C38-11D5-6FCF-BB3200333F07
</location>
</link>
</source>
<target>
<name> LATravelAgency </name>
<link type = "uddi-location">
<location>http://www.latravelagency.com/uddi?
4C9DADD0-5C69-11D5-2FCF-BB4500334F02
</location>
</link>
</target>
</exclusion>
</bbr>
<wsrl>
```

List 6. The example of WSRL between web services.

```
<?xml version = "1.0"?>
<!DOCTYPE wsrl SYSTEM "wsrl.dtd">
<wsrl>
<bbr>
<partnership id = p1>
<source>
<name> AZCompany.com </name>
<link type = "uddi-location">
<location>http://www.azcompany.com/uddi?
4C9DADD0-5C38-11D5-6FCF-BB3200335F07
</location>
</link>
</source>
<target>
<name> AZTravelAgency </name>
<link type = "uddi-location">
<location>http://www.aztravelagency.com/uddi?
4C9DADD0-5C39-11D5-3FCF-BB4500333F88
</location>
</link>
</target>
</partnership>
</bbr>
<wsrl>
```

the appropriate services for the business process to be composed.

The Figure 8 screen image shows the user interface of the WSOM for where a customer to specify business preferences.

The following steps are needed to input their business preferences.

(1) Select a service cluster from the expected service list;
(2) Specify service clusters to follow for each selected service cluster; according to that information, the WSOM is able to construct an expected business process diagram, e.g., as shown in Figure 6;
(3) Specify preferences for the service cluster; at present, the preferences include Cost, Time, Benefit, Quality and possibly abstract services listed under "Included Services".

The Figure 9 screen image shows how customers input business rules into the WSOM.

The following steps are needed to input business rules into the WSOM:

(1) specify types of business rules in the list of Types;
(2) specify the rule related to service clusters in the Relevant Service list;
(3) specify the priority of the rule;
(4) specify the condition of the rule; in general, the conditions contain the information on service preferences, such as Cost, Time and so on;
(5) Specify the corresponding behavior of a rule under a certain condition.

In addition, a sample implementation the paper provides comprises two mechanisms to capture the business requirements from the customers, namely, (1) GUI based BPOL Analyzer, and (2) program-based BPOL converter. The GUI based BPOL analyzer is part of an Eclipse based version of WSOM, providing a set of interactive interfaces to capture business requirements, preferences, business rules and then automatically generate XML-based BPOL requirement document.

As shown in the Figure 10 screen image, the WSOM also provides a wizard to capture the requirements by selecting the "New Expected Service" menu item to create a new entry, the "New BPOL" menu item to create a new BPOL document or the "Open BPOL" menu item to open an existing BPOL, which can then be modified by adding new entries.

As shown in Figure 11, the business requirements captured by WSOM can be configured to generate execution language documents such as WSFL, which is the produced by the sample implementation. Also, note the WSOM can be adapted to generate BPEL through the use of an output adaptor created for BPEL.

At the same time, the decision maker or customer can monitor the entire process. If necessary, the customer can use the interface to manually modify the result or tune up the constructed business process.

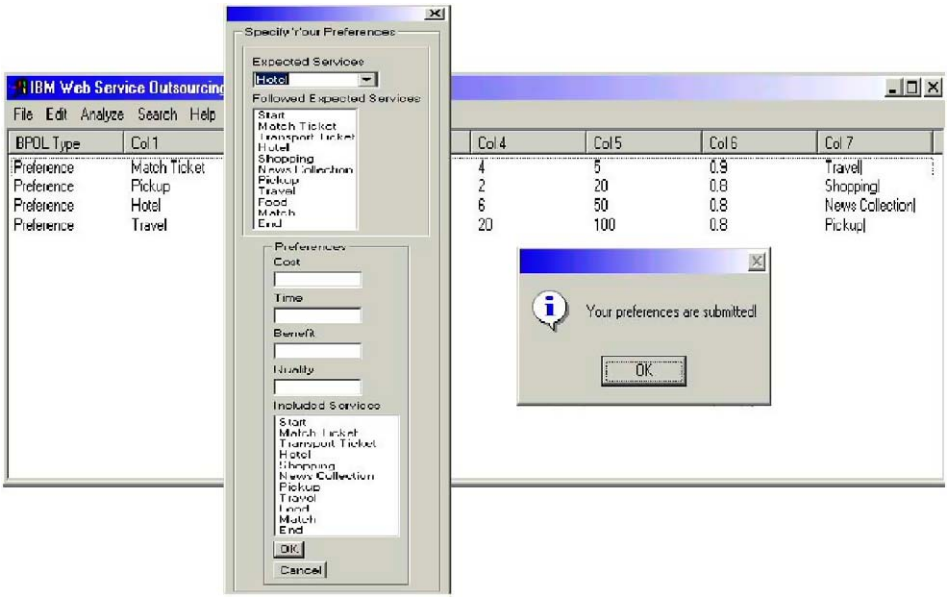


Figure 8. Customer requirement input screen.

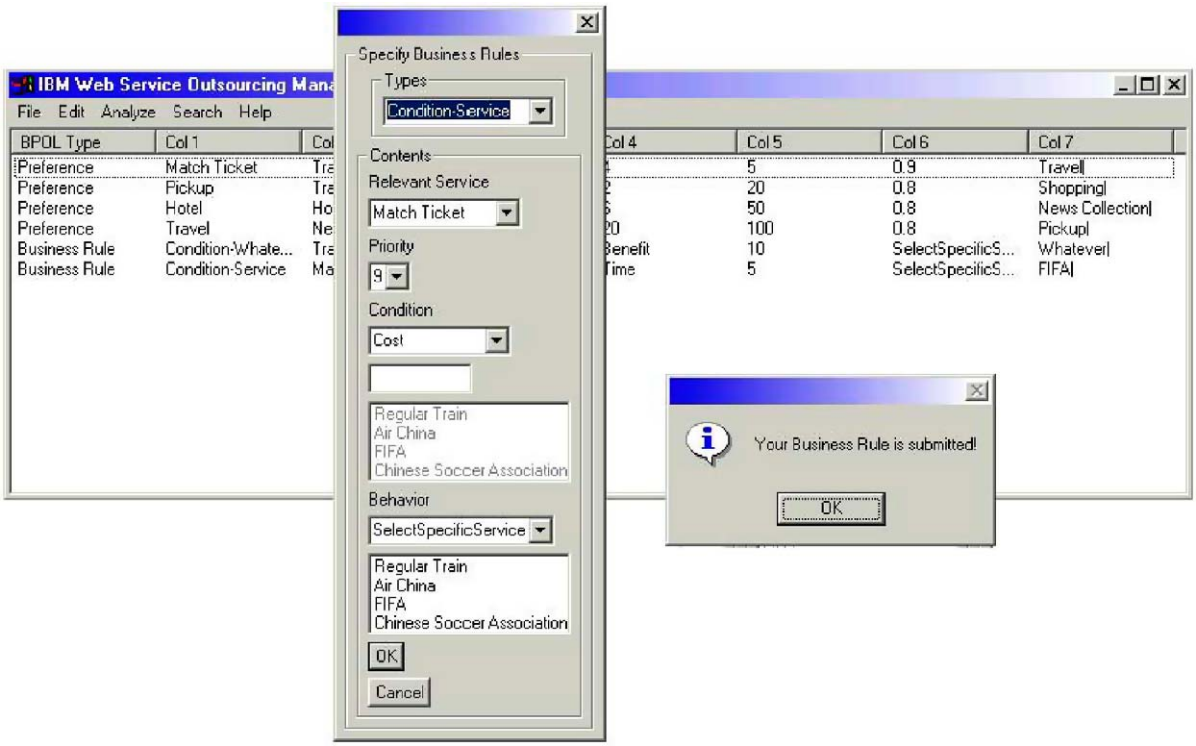


Figure 9. Business rules input screen.

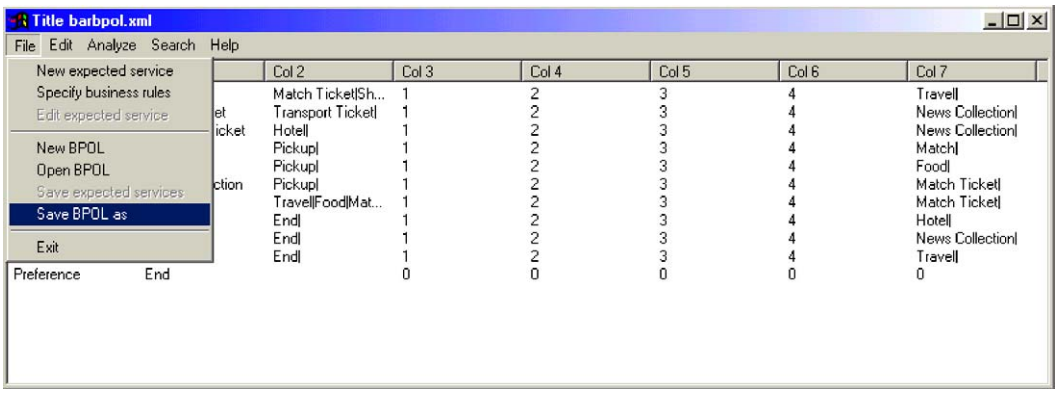


Figure 10. BPOL generation.

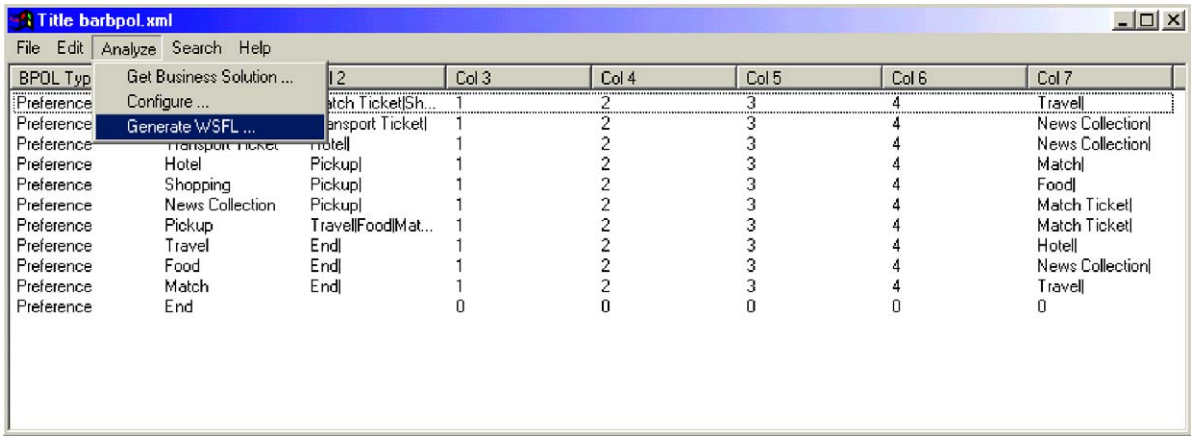


Figure 11. Web services flow sketch generation.

For example, the resulting business process obtained by the WSOM for customers can be further adjusted based on changing conditions. As the business process evolves, changes will need to be made. Therefore, the run-time execution behaviors of an automatically constructed business process must be carefully monitored, evaluated and adapted for the expected quality of service. Customers might encounter changes in the business plan, or find that the initial information supplied to the WSOM is insufficient. Therefore, it is essential to provide customers with capabilities to tune the generated business processes. After the tuning, the WSOM will provide a new process analysis based on the updated information the new performance information for customers to evaluate. Thus, a final optimum business process can be obtained.

Table 1 lists the sample Web services clusters WS1 to WS9 in the first column "Web services cluster". The second column, "Description", provides a short description of what the Basketball fan is looking for from the cluster. The candidate Web services are shown in the third column "Candidate Services". These candidates were discovered from UDDI registries or WS-Inspection documents, as shown in Figure 3 where Advanced Web Services Discovery Engine queries UDDI registry and develops an available service list. In general, the WSOM searches the UDDI registry so as to narrow down the scope of possible service candidates by business names, categories and so on, if specified in the business requirements. After that filtering process, the resulting list of Web services are invoked by the WSOM to obtain real values such as Cost, Time and so forth. Table 1 shows the Cost (in US dollars) and Time (in Days) in the fourth and fifth columns.

In Table 1, the service value is calculated through the formula (8). Please note that the value of R_i^{d} is equal to 0, which means that customers expect to get the minimum of the combination of Cost and Time. From the WSOM, the final business process is constructed by the following service set:

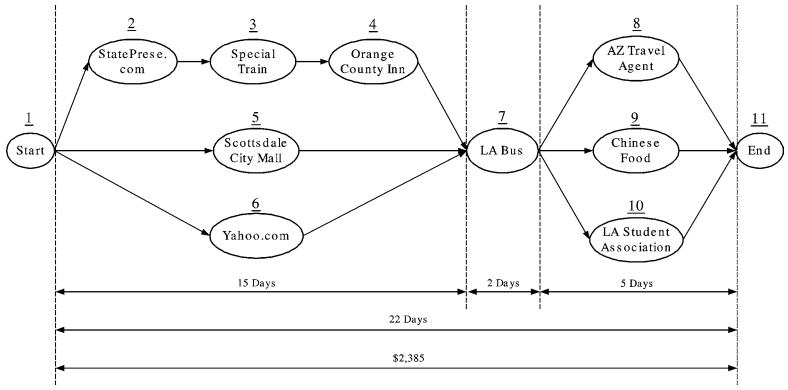


Figure 12. The optimized basketball fan's travel process.

$$S^* = \{0010, 001000, 01, 010, 001, 001, 10, 100, 010\}. \quad (13)$$

In addition, this service set contains nine sets of binary numbers with each set presenting a Web services clusters, i.e., WS1 through WS9 as shown in Table 1. Each binary number contains a digit for each candidate service, and a single "1" for the candidate service selected. Also note that within each Web services cluster normalized values for Cost and Time (the sixth and seventh columns of Table 1 are determined by giving the maximum value a normalized value of unity (1.0)). Weighted values for Cost and Time (shown in the eighth and ninth columns of Table 1) are determined by a calculation that gives desired weights, in accordance with the service value metric shown in Equation (8). This calculation may be understood from the following example, using Equation (8). With reference to Figure 6 and Table 1, assume an expected traveling time of zero (0) days and an expected expense of zero (0) dollars. In that case, $R_i^{\mathrm{d}} = 0$ ($i = 1, 2$). Here R_1^{d} is the expected days specified in the customer requirement annotation data (e.g., BPOL). R_2^{d} is the expected expense in this trip. Therefore the service value can be calculated in the following simpler way:

$$\mathit{ServiceValue} = \sum_{i=1}^{i=P} E_i = \sum_{i=1}^{P} \left[\frac{1}{2} w_i (R_i^*)^2 \right]. \quad (14)$$

In the example shown in Table 1, take the first service "TempTicket.com" as an example, the normalized cost $= 400/500 = 0.8$; the normalized time is $3/10 = 0.3$. Since we just picked up two requirement indicators (Cost and Time), then we can set $P = 2$ in the Service Value equation. That is

$$\mathit{ServiceValue} = \sum_{i=1}^{i=P} E_i = \sum_{i=1}^{2} \left[\frac{1}{2} w_i (R_i^*)^2 \right] = \frac{1}{2} w_1 (R_1^*)^2 + \frac{1}{2} w_2 (R_2^*)^2. \quad (15)$$

In Table 1, the weighted cost and weighted time correspond to $(1/2)w_1(R_1^*)^2$ and $(1/2)w_2(R_2^*)^2$ respectively.

In the example shown in Table 1, the weighted cost of the first service "TempTicket.com" is 0.192. This resulted from the following calculation process:

$$\frac{1}{2} w_1 (R_1^*)^2 = \frac{1}{2} \cdot 0.6 \cdot (0.8)^2 = 0.192. \quad (16)$$

The weighted time of the same service is 0.018 derived from the following calculation process:

$$\frac{1}{2} w_2 (R_2^*)^2 = \frac{1}{2} \cdot 0.4 \cdot (0.3)^2 = 0.018. \quad (17)$$

After getting the optimum result, using an optimization algorithm such as GA, the final business requirement diagram corresponding to (13) may be shown with reference to Figure 12.

Figure 12 is very similar to Figure 6, with the nine Web services cluster replaced by the name of the corresponding selected actual services. Furthermore, Figure 12 is annotated to show two indicators, Cost and Time. As usual, there needs to be a balance between the Cost and the Time when calculating the Service Value (the tenth column in Table 1). Different weights on indicators should be taken into consideration when obtaining the optimum service selection from all available services in a UDDI registry. The final Service Value is calculated based on the combination of normalized and weighted Cost and Time, or some other suitable measurement that are suitable.

7. Related Work

Paper [12] addresses a particular subset of automatic or semi-automatic composition of existing Web Services. The toolkit, SWORD, is a set of tools for the composition of a class of Web Services including "information-providing" services. It focuses on information integration using its own service model rather than the discovery based dynamic Web Services composition.

The Service Community proposed in [13] defines a set of interfaces to enforce service providers to implement the full or partial interface. The concept of Web Services Cluster proposed in this paper refers to a set of services that perform a specific task. It is task-oriented concept, which may have the same service interface or different service interface. Further, there are no business requirements used in [13] during the composition process. So the constructed business process may not the optimal one.

The service merging techniques presented in [14] are based on the composition of services with the intended goal that, for a given request, a composite service expands into a ready-to-use optimal successful plan. It is a rule based dynamic flow template composition mechanism rather than the real service binding framework based on service discovery and capability matching presented in this paper. Again, the universal representation of customers' requirements is not addressed.

The Web Services Policy Framework (WS-Policy) was created to represent and communicate the policies of a Web Service. It covers some aspects of service requirements, preferences, and capabilities [22]. BPOL presented in this paper describe the business requirements and services relationships which are used for composing a business process. Therefore BPOL is an upper level policy representation which could link to WS-Policy which is used for description a Web service. Moreover, the current WS-Policy specification [22] concentrates on security policies only.

Using regular XML file (e.g., BPOL) or Resource Definition Framework (RDF) [25] or OWL-S [24] will have different impacts on the degrees of automation process for discovery, selection, and composition of Web services. In fact, capturing the specific semantics about the business requirements and operations is the most important thing. Based on our industry solution experience, we think full automation for complicated business process integration and collaboration is still a dream. A practical way is to define and leverage business semantics to integrate people, process and information [26]. That is the reason why this paper concentrates on the contents defined in BPOL and specific optimization model and algorithm for services composition by implementing GUI-based WSOM tools that help people get involved in the semi-automated approach.

8. Conclusions

In summary, we have presented a mathematical model to automate the process of dynamic composition of a business process using Web Services to match customers' requirements. The requirements are analyzed and optimized to generate proposed XML-based BPOL requirement documents for business process outsourcing. Based on BPOL, a two-level service selection mechanism is proposed to configure a new business process based on customer's requirements. The first-level service selection mechanism is performed by the advanced Web Services Discovery Engine of the WSOM, whose search criteria is created automatically by BPOL processor, to narrow down the available service list. Then the second-level service selection mechanism is performed by a global optimization algorithm, Genetic Algorithm, to perform the capability matching and then generate the best business process to satisfy the customers' requirements defined in BPOL. Furthermore, relationships such as partnerships, alliances, competitors among the services are also taken into account when select the final services as part of the optimization process. Lastly, the generated business process flow can be further adjusted and tuned as business requirements changes and evolve.

The research prototype of Web Services Outsourcing Manager presented in this paper has been released and updated on IBM alphaWorks [19] since Sept 2002. The released Web Services Outsourcing Manager is a framework that enables dynamic composition of Web service flow based on customer requirements. The customer requirements are analyzed and optimized to generate an annotation document for business process outsourcing. This service-oriented architecture allows effective searching for appropriate Web services and integration of them into one composite Web service for performing a specific task. Meanwhile, it provides a seamless, integrated framework for composition of template-based business processes and event-driven business processes. We have been working on extending the WSOM framework to support business process

integration and management in Grid environment by leveraging the latest Web services and Grid specifications such as WS-Resource Framework [17] and BPEL4WS [9].

Finally, we envision the following issues can be further explored to enhance dynamic composition of business process:

- Creating more business semantics for assisting dynamic service discovery and selection based on real business scenarios;
- Investigating the pricing model for dynamic business process composition and outsourcing;
- Modeling the Web services discovery and selection process using PetriNet [10] or system dynamics methods [11];
- Implementing a dynamic business process composition scenario in real Grid computing solution rather than this simple travel planning scenario.

Acknowledgements

This paper is the extension of a conference paper [20]. The major ideas presented in this paper has been presented in the keynote speech at the Second International Workshop on Grid and Cooperative Computing (GCC 2003) held in Shanghai on December 7–10, 2003. We would like to thank Henry Chang, Jen-Yao Chung and John Y. Sayah for their support for this work. We also like to send our thanks to Tian Chao and Qun Zhou for their reviewing and editing partial contents presented in this paper.

References

1. R. Schmelzer et al., *XML and Web Services Unleashed*. SAMS Publishing, 2002.
2. S. Aissi, P. Malu and K. Srinivasan, "E-business Process Modeling: The Next Big Step", *IEEE Computer*, Vol. 35, No. 5, pp. 55–62, May 2002.
3. L.-J. Zhang, T. Chao, H. Chang and J.-Y. Chung, "XML-Based Advanced UDDI Search Mechanism for B2B Integration", *Electronic Commerce Research*, Vol. 3, Nos. 1–2, 2003, pp. 25–42.
4. "Business Explorer for Web Services (BE4WS)", *IBM AlphaWorks*, 2001, www.alphaworks.ibm.com/tech/be4ws
5. L.-J. Zhang, H. Chang and T. Chao, "Web Services Relationships Binding for Dynamic e-Business Integration", in *International Conference on Internet Computing (IC'02)*, Las Vegas, May 24–27, 2002.
6. M.D. Vose, *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press: Cambridge, MA, 1999.
7. "Web Services Outsourcing Manager", *IBM AlphaWorks*, September 2002, http://www.alphaworks.ibm.com/tech/wsom
8. L.-J. Zhang, Z.-H. Mao and Y.-D. Li, "Mathematical Analysis of Mutation Operator in Generic Algorithms and Its Improved Strategy", in *International Conference on Neural Computing*, Beijing, 1995.
9. "Business Process Execution Language (BPEL4WS, Version 1.1)", May 2003, http://xml.coverpages.org/BPELv11-May052003Final.pdf
10. J.L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice-Hall: Englewood Cliffs,1981.
11. C.W. Kirkwood, "System Dynamics Methods: A Quick Introduction", 1998, http://www.public.asu.edu/~ kirkwood/sysdyn/SDIntro/SDIntro.htm
12. S.R. Ponnekanti and A. Fox, "SWORD: A Developer Toolkit for Building Composite Web Services", Standford University, 2002, http://swig.stanford.edu/pub/publications/sword/www11.pdf
13. B. Benatallah et al., "Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services", in *IEEE International Conference on Data Engineering'2002*, pp. 297–308.
14. R.M. Fadel, "Integrating Web Services by Composition", December 2001, http://www-cs-students.stanford.edu/~rfadel/Papers/IntWebServ.pdf
15. I. Foster, C. Kesselman, J.M. Nick and S. Tuecke, "Grid Services for Distributed System Integration", *Computer*, Vol. 35, No. 6, pp. 37–46, 2002.
16. "OGSA (Open Grid Services Architecture)", 2002, http://www.globus.org/ogsa/
17. "Web Services Notification and Web Services Resource Framework (WSRF)", 2004, http://www-106.ibm.com/developerworks/webservices/library/ws-resource/
18. L.-J. Zhang, J.-Y. Chung and Q. Zhou, "Developing Grid Computing Applications, Part 1", *IBM DeveloperWorks*, 2002, http://www-106.ibm.com/developerworks/grid/library/gr-grid1/ (Discover Grid Computing, *developerWorks Journal*, February 2003, pp. 14–19.)
19. "Web Services Outsourcing Manager (WSOM)", *IBM AlphaWorks*, September 30, 2002, http://www.alphaworks.ibm.com/tech/wsom
20. L.-J. Zhang, B. Li, T. Chao and H. Chang, "On Demand Web Services-Based Business Process Composition", in *Proceedings of the 2003 IEEE Conference on System, Man, and Cybernetics (SMC'03)*, October 2003, pp. 4057–4064.
21. G. Wasson and M. Humphrey, "Toward Explicit Policy Management for Virtual Organizations", in *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks (POLICY'03)*, 4–6 June 2003, pp. 173–182.
22. D. Box, F. Curbera, M. Hondo, C. Kale, D. Langworthy, A. Nadalin, N. Nagaratnam, M. Nottingham, C. von Riegen and J. Shewchuk, "Specification: Web Services Policy Framework (WSPolicy)", 28 May 2003, http://www-106.ibm.com/developerworks/library/ws-polfram/
23. "Eclipse (A Universal Tool Platform)", eclipse.org
24. "OWL-S: Semantic Markup for Web Services", http://www.daml-s.org/owl-s/1.0/owl-s.html
25. "Resource Description Framework (RDF)", http://www.w3.org/RDF/
26. J. Sayah and L.-J. Zhang, "On-Demand Business Collaboration Enablement With Web Services", IBM Research Report No. RC22926(W0309-191), September 30, 2003, IBM T.J. Watson Research Center. To appear in *Decision Support System*.
27. K. Ballinger et al., "Web Services Inspection Language (WS-Inspection)", 2001, http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html