

# Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE)

by

Saleh Saeed AlZahrani

Software Technology Research Laboratory  
Faculty of Technology  
De Montfort University, Leicester, UK

This thesis is submitted in partial fulfillment  
of the requirements for the degree of

**Doctor of Philosophy**

Computer Sciences

2010

# Abstract

e-Learning is becoming an influential role as an economic method and a flexible mode of study in the institutions of higher education today which has a presence in an increasing number of college and university courses. e-Learning as system of systems is a dynamic and scalable environment. Within this environment, e-learning is still searching for a permanent, comfortable and serviceable position that is to be controlled, managed, flexible, accessible and continually up-to-date with the wider university structure. As most academic and business institutions and training centres around the world have adopted the e-learning concept and technology in order to create, deliver and manage their learning materials through the web, it has become the focus of investigation. However, management, monitoring and collaboration between these institutions and centres is limited.

Existing technologies such as grid, web services and agents are promising better results. In this research a new architecture has been developed and adopted to make the e-learning environment more dynamic and scalable by dividing it into regional data grids which are managed and monitored by agents. Multi-agent technology has been applied to integrate each regional data grid with others in order to produce an architecture which is more scalable, reliable, and efficient. The result we refer to as Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE).

Our RDADeLE architecture is an agent-based grid environment which is composed of components such as learners, staff, nodes, regional grids, grid services and Learning Objects (LOs). These components are built and organised as a multi-agent system (MAS) using the Java Agent Development (JADE) platform. The main role of the agents in our architecture is to control and monitor grid components in order to build an adaptable, extensible, and flexible grid-based e-learning system.

Two techniques have been developed and adopted in the architecture to build LOs' information and grid services. The first technique is **the XML-based Registries Technique (XRT)**. In this technique LOs' information is built using XML registries to be discovered by the learners. The registries are written in Dublin Core Metadata Initiative (DCMI) format. The second technique is **the Registered-based Services Technique (RST)**. In this technique the services are grid services which are built using agents. The services are registered with the Directory Facilitator (DF) of a JADE platform in order to be discovered by all other components. All components of the RDADeLE system, including grid service, are built as a multi-agent system (MAS). Each regional grid in the first technique has only its own registry, whereas in the second technique the grid services of all regional grids have to be registered with the DF.

We have evaluated the RDADeLE system guided by both techniques by building a simulation of the prototype. The prototype has a main interface which consists of

the name of the system (RDADeLE) and a specification table which includes Number of Regional Grids, Number of Nodes, Maximum Number of Learners connected to each node, and Number of Grid Services to be filled by the administrator of the RDADeLE system in order to create the prototype.

Using **the RST technique** shows that the RDADeLE system can be built with more regional grids with less memory consumption. Moreover, using **the RST technique** shows that more grid services can be registered in the RDADeLE system with a lower average search time and the search performance is increased compared with **the XRT technique**. Finally, using one or both techniques, **the XRT** or **the RST**, in the prototype does not affect the reliability of the RDADeLE system.

# Declaration

I declare that the work described in this thesis is original work undertaken by me for the degree of Doctor of Philosophy, at the Faculty of Technology, De Montfort University, Leicester, United Kingdom.

No part of the material described in this thesis has been submitted for the award of any other degree or qualification in this or any other university or college of advanced education.

This thesis is written by me and produced using L<sup>A</sup>T<sub>E</sub>X

**Saleh S. AlZahrani**

Leicester, United Kingdom. 2010

# List of Publications

## Published papers:

1. Saleh AlZahrani, Aladdin Ayesh, and Hussein Zedan. Multi-agent Based Dynamic e-Learning Environment. *International Journal of Information Technology and Web Engineering (IJITWE)*. Vol. 4, Issue 2. ITJ5072. June 2009.
2. Saleh AlZahrani, Aladdin Ayesh, and Hussein Zedan, Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE), Conference Paper, *IEEE Proceeding HSI'08*, 1-4244-1543, Krakow, Poland 2008.
3. Saleh AlZahrani, Aladdin Ayesh, and Hussein Zedan, Multi-Agent System Based Regional Data Grid, Conference Paper, *IEEE Proceeding ICCES'08*, 1-4244-2116, Cairo, Egypt 2008.

## Dedication

***To my mother and father:***

*To my father who passed away before seeing me complete my PhD and my mother  
who taught me much and is waiting in anticipation for my PhD,  
for their endless love.*

***To my family:***

*Who have supported me and shown understanding and patience during the research  
years.*

## Acknowledgements

First and foremost, my deepest gratitude is for ALLAH for all his blessings, without which my work would not have been possible.

I would also like to express my sincere gratitude for my supervisor Dr. Aladdin Ayesh at De Montfort University, Leicester, UK, for his guidance, ideas, valuable direction and support throughout my research study. I am deeply indebted to him for his insights and suggestions about the thesis topic and for valuable supervision at various stages of my work. He has helped me in many ways and has developed my academic thinking, problem solving and technical writing which will be very helpful in my future work and research.

A grateful acknowledgement should be mentioned for Prof. Hussein Zedan at De Montfort University, Leicester, UK, for his ideas, encouragement and support. His support has greatly inspired me to complete this work in the required time.

I would also like to thank my sponsor Royal Commission For Jubail and Yanbu (RCJY) in Saudi Arabia, especially Dr. Musleh AlOtaibi, Director General of the Royal Commission in Jubail.

Special thanks to my wife for her great support and patience. She had to sacrifice a lot of time and endure with patience to help me through difficulties during the research period.

I am grateful for the support and assistance that I have received from all members



of the STRL research groups in DMU, especially, Dr. Omar Al-Dabbas. I have benefitted from many discussions and collaboration with them.

Last, but most certainly not least, I would like to express my gratefulness to my friends and colleagues who helped me to overcome many troubles and supported me during the research.

Saleh S. AlZahrani

## List of Acronyms

<b>AA</b>	Administrative Agent
<b>AAII</b>	Australian Artificial Intelligent Institute
<b>AAP</b>	April Agent Platform
<b>ABLE</b>	Agent Building and Learning Environment
<b>ACL</b>	Agent Communication Language
<b>ADK</b>	Agent Development Kit
<b>ADL</b>	Advanced Distributed Learning
<b>AICC</b>	The Aviation Industry CBT Committee
<b>AMS</b>	Agent Management System
<b>ANM</b>	Agent Network Manager
<b>ANS</b>	Agent Name Server
<b>AOD</b>	Agent-Oriented Development
<b>AOP</b>	Agent-Oriented Programming
<b>AOSE</b>	Agent-Oriented Software Engineering
<b>API</b>	Application Program Interface
<b>ARIADNE</b>	the Alliance of Remote Instructional Authoring and Distribution Networks for Europe
<b>ASA</b>	Agent Service Adapter
<b>AUML</b>	Agent Unified Modeling Language
<b>BDI</b>	Belief-Desire-Intention (typically of agent architectures)
<b>CBT</b>	Computer-Based Training
<b>CCLRC</b>	Council for the Central Laboratory of the Research Councils
<b>CERN</b>	European Organisation for Nuclear Research
<b>CGSP</b>	China Grid Support Platform
<b>CPU</b>	Central Processing Unit
<b>CT</b>	Container Table
<b>DCMI</b>	Dublin Core Meta-data Initiative
<b>DESIRE</b>	Design and Specification of Interacting Reasoning components
<b>DF</b>	Directory Facilitator
<b>DFS</b>	Distributed File System
<b>DNS</b>	Domain Name System
<b>ECAR</b>	Educause Center for Applied Research
<b>EGEE</b>	Enabling Grid for E-Science
<b>EPL</b>	Eclipse Public License
<b>EPSRC</b>	Engineering and Physical Sciences Research Council
<b>ESESGrid</b>	Engineering Structure Experiment and Simulation Grid
<b>FIPA</b>	Foundation for Intelligent Physical Agents
<b>FIPA-OS</b>	Foundation for Intelligent Physical Agents - Open Source

<b>GADT</b>	Global Agent Descriptor Table
<b>GARA</b>	General-purpose Architecture for Reservation and Allocation
<b>GGF</b>	Global Grid Forum
<b>GIG</b>	Grid Infrastructure Group
<b>GIS</b>	Grid Information Service
<b>GPEL</b>	Grid Process Execution Language
<b>GPFS</b>	General Parallel File System
<b>GPL</b>	GNU General Public License
<b>GRAM</b>	Grid Resource Access and Management Protocol
<b>GridFTP</b>	Grid File Transfer Protocol
<b>GRIP</b>	Grid Resource Information Protocol
<b>GSML</b>	Grid Service Markup Language
<b>HTTP</b>	Hypertext Transfer Protocol
<b>ICMP</b>	Internet Control Message Protocol
<b>IEEE</b>	Electronic and Electronic Engineers
<b>IJITWE</b>	International Journal of Information Technology and Web Engineering
<b>IMS</b>	Instructional Management Standards
<b>J2ME</b>	Java 2 Platform, Micro Edition
<b>JADE</b>	Java Agent Development Framework
<b>JAS</b>	Java Agent Service
<b>JATLite</b>	Java Agent Template, Lite
<b>JISC</b>	Joint Information Systems Committee
<b>JNDI</b>	Java Naming and Directory Interface
<b>JVM</b>	Java Virtual Machine
<b>KIF</b>	Knowledge Interchange Format
<b>KQML</b>	Knowledge Query Manipulation Language
<b>KSA</b>	Kingdom of Saudi Arabia
<b>LA</b>	Learner Agent
<b>LAN</b>	Local Area Network
<b>LCMM</b>	Life Cycle Management Model
<b>LDAP</b>	Lightweight Directory Access Protocol
<b>LEAP</b>	Lightweight Extensible Agent Platform
<b>LGPL</b>	GNU Lesser General Public License
<b>LMS</b>	Learning Management Systems
<b>LO</b>	Learning Object
<b>LOM</b>	IEEE Learning Object Meta-data
<b>LTSC</b>	Learning Technology Standard Committee
<b>MAF</b>	Mobile Agent Facility
<b>MAS</b>	Multi-Agent System
<b>MaSE</b>	The Multi-agent System Engineering
<b>MIDP</b>	Mobile Information Device Profile

<b>MIPS</b>	Million Instructions Per Second
<b>MTP</b>	Message Transport Protocol
<b>MVC</b>	Model,View Controller
<b>NA</b>	Node Agent
<b>NFS</b>	Network File System
<b>NGS</b>	National Grid Service
<b>NWICG</b>	Northwest Indiana Computational Grid
<b>OAA</b>	Open Agent Architecture
<b>OGF</b>	Open Grid Forum
<b>OGSA</b>	Open Grid Service Architecture
<b>OGSI</b>	Open Grid Services Infrastructure
<b>OMG</b>	Object Management Group
<b>OMT</b>	Object-Modeling Technique
<b>OOD</b>	Object-Oriented Development
<b>OOP</b>	Object-Oriented Programming
<b>ORB</b>	Object Request Broker
<b>P2P</b>	Peer to Peer
<b>PDA</b>	Personal Digital Assistant
<b>RA</b>	Regional Agent
<b>RDADeLE</b>	Regionally Distributed Architecture for Dynamic e-Learning Environment
<b>RDF</b>	Resource Description Format
<b>RMA</b>	Remote Monitoring Agent
<b>RMI</b>	Remote Method Invocation
<b>ROADMAP</b>	Role Oriented Analysis and Design for Multi-Agent Programming
<b>RST</b>	Registered-based Service Technique
<b>SA</b>	Service Agent
<b>SAML</b>	Security Assertion Markup Language
<b>SCORM</b>	Sharable Content Object Reference Model
<b>SDSS</b>	Sloan Digital Sky Survey
<b>SIRENE</b>	Sharing Infrastructure and Resources in Europe
<b>SL</b>	Semantic Language
<b>SMTP</b>	Simple Mail Transfer Protocol
<b>SOAP</b>	Simple Object Access Protocol
<b>SPEC</b>	Standard Performance Evaluation Corporation)
<b>SQL</b>	Structured Query Language
<b>STFC</b>	Science and Technology Facilities Council
<b>SWAP</b>	Simple Work-flow Access Protocol
<b>TCP/IP</b>	Transmission Control Protocol/Internet Protocol
<b>UDDI</b>	Universal Description, Discovery and Integration
<b>UDP</b>	User Datagram Protocol

<b>UML</b>	Unified Modeling Language
<b>URI</b>	Uniform Resource Identifier
<b>VPN</b>	Virtual Private Network
<b>W3C</b>	World Wide Web Consortium
<b>WAN</b>	Wide Area Network
<b>WG</b>	Work Group
<b>WS-Addressing</b>	Web Services Addressing
<b>WS-BPEL</b>	Web Services - Business Process Execution Language
<b>WSCI</b>	Web Service Choreography Interface
<b>WSDL</b>	Web Service Description Language
<b>WSFL</b>	Web Service Flow Language
<b>WSInspection</b>	Web Services Inspection Language
<b>WSRF</b>	Web Services Resource Framework
<b>WS-Security</b>	Web Services Security
<b>WUN</b>	Worldwide Universities Network
<b>WWW</b>	World Wide Web
<b>XACML</b>	Extensible Access Control Markup Language
<b>XML</b>	Extensible Markup Language
<b>XRT</b>	XML-based Registries Technique

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Declaration</b>	<b>v</b>
<b>List of Publications</b>	<b>vi</b>
<b>Dedication</b>	<b>vii</b>
<b>Acknowledgements</b>	<b>viii</b>
<b>List of Acronyms</b>	<b>x</b>
<b>List of Tables</b>	<b>xxiii</b>
<b>List of Figures</b>	<b>xxiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation and Significance . . . . .	1
1.2 Problem Statement . . . . .	3
1.2.1 Research Question . . . . .	5
1.3 Research Contributions . . . . .	7
1.4 Research Methodology . . . . .	8
1.5 Thesis Organisation . . . . .	9
<b>2 Grid Computing</b>	<b>10</b>
2.1 Introduction . . . . .	10
2.2 How Grid Computing Works . . . . .	11
2.3 Characteristics of Grid Computing . . . . .	12
2.4 Grid Category . . . . .	14

2.4.1	Computational Grid . . . . .	14
2.4.2	Data Grid . . . . .	15
2.4.3	Cluster Grids . . . . .	17
2.4.4	Enterprise Grids . . . . .	17
2.4.5	Extraprise Grids . . . . .	17
2.4.6	Global Grids . . . . .	18
2.5	Grid Architecture . . . . .	18
2.6	Data Grid Applications . . . . .	21
2.6.1	UK National Grid Service (NGS) . . . . .	21
2.6.2	The Sakai Project . . . . .	22
2.6.3	Enabling Grid for E-ScienceE (EGEE) Project . . . . .	22
2.6.4	Sloan Digital Sky Survey (SDSS) Project . . . . .	22
2.6.5	TeraGrid . . . . .	23
2.7	Web and Grid Services . . . . .	23
2.8	Grid Service Tools . . . . .	27
2.8.1	Middleware Tools . . . . .	28
2.8.1.1	Globus Toolkit . . . . .	28
2.8.1.2	JC-Grid . . . . .	28
2.8.1.3	Grid-Gain . . . . .	29
2.8.1.4	Jini-Middleware . . . . .	29
2.8.2	Languages for Grid Services Specification and Composition . .	29
2.8.3	Grid Simulators . . . . .	32
2.8.3.1	MicroGrid . . . . .	32
2.8.3.2	Gridsim . . . . .	33

2.8.3.3	Simgrid . . . . .	35
2.9	Review of Data Grid Technology towards E-learning Requirements . .	37
2.10	Summary . . . . .	38
<b>3</b>	<b>Agents-based Computing</b>	<b>39</b>
3.1	Introduction . . . . .	39
3.2	Agent Technology Overview . . . . .	40
3.2.1	Characteristics of Agents . . . . .	41
3.2.2	Classifications of Software Agents . . . . .	42
3.2.3	Main Agent Architectures . . . . .	44
3.2.3.1	Reactive Architecture . . . . .	46
3.2.3.2	Cognitive Architecture . . . . .	46
3.2.3.3	Hybrid Architecture . . . . .	47
3.2.4	Agent Communications . . . . .	47
3.2.4.1	Speech Acts . . . . .	48
3.2.4.2	Knowledge Query and Manipulation Language (KQML)	48
3.2.4.3	Knowledge Interchange Format (KIF) . . . . .	49
3.2.4.4	FIPA-Agent Communication Language (FIPA-ACL) .	50
3.2.4.5	Ontologies . . . . .	50
3.2.5	Agent Interaction Protocols . . . . .	51
3.2.5.1	Coordination Protocols . . . . .	51
3.2.5.2	Cooperation Protocols . . . . .	51
3.2.5.3	Contract Net . . . . .	51
3.2.5.4	Market Mechanisms . . . . .	52
3.2.5.5	Blackboard Systems . . . . .	52



3.2.5.6	Negotiation . . . . .	53
3.3	Agent Design and Development Methodologies and Languages . . . . .	53
3.3.1	Terminology . . . . .	53
3.3.2	Agent-oriented Methodologies . . . . .	54
3.3.2.1	Gaia . . . . .	54
3.3.2.2	Prometheus . . . . .	55
3.3.2.3	Tropos . . . . .	56
3.3.2.4	Multi-agent System Engineering(MaSE) . . . . .	56
3.3.2.5	Role Oriented Analysis and Design for Multi-Agent Programming(ROADMAP) . . . . .	57
3.3.3	Object-oriented Based Methodologies . . . . .	57
3.3.3.1	Australian Artificial Intelligent Institute (AAIL) . . . . .	57
3.3.3.2	Agent Unified Modeling Language (AURL) . . . . .	58
3.3.4	Other Methodologies . . . . .	58
3.3.4.1	(Design and Specification of Interacting Reasoning Com- ponents(DESIKE)) . . . . .	58
3.3.4.2	MAS-CommonKADS . . . . .	59
3.3.5	Programming Languages . . . . .	59
3.3.6	AOP Versus OOP . . . . .	60
3.4	Multi-agent Systems (MAS) . . . . .	61
3.4.1	Characteristics of Multi-agent Environments . . . . .	61
3.4.2	Applications of Multi-agent Systems (MAS) . . . . .	61
3.5	Agents Role in Grid Computing . . . . .	62
3.5.1	Grid Resource Discovery . . . . .	63
3.5.2	Data Acquisition and Retrieval . . . . .	63

3.5.3	Provision Internal Processing in Grid Environment . . . . .	64
3.6	Agent Platforms and Simulators . . . . .	64
3.6.1	IBM Aglets Workbench . . . . .	65
3.6.2	Concordia . . . . .	65
3.6.3	Odyssey . . . . .	66
3.6.4	Voyager . . . . .	66
3.6.5	JATLite . . . . .	67
3.6.6	Agent Development Kit . . . . .	68
3.6.7	April Agent Platform . . . . .	69
3.6.8	Comtec Agent Platform . . . . .	69
3.6.9	FIPA-OS . . . . .	69
3.6.10	Grasshopper . . . . .	70
3.6.11	JACK Intelligent Agents . . . . .	70
3.6.12	JAS (Java Agent Services API) . . . . .	71
3.6.13	ZEUS . . . . .	71
3.6.14	JADE Platform . . . . .	72
3.6.15	Lightweight Extensible Agent Platform (LEAP) . . . . .	74
3.6.16	Agent Platforms Comparison . . . . .	75
3.7	Review of Agent Technology towards E-learning Requirements . . . .	76
3.8	Summary . . . . .	77
<b>4</b>	<b>Architectural Design of Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE)</b>	<b>80</b>
4.1	Introduction . . . . .	80
4.2	Requirement Analysis . . . . .	81

4.2.1	Previous e-Learning Architectures . . . . .	82
4.2.2	Context and Motivation . . . . .	84
4.2.3	Requirements for Dynamic RDADeLE Environment . . . . .	89
4.2.4	The Architectural Data Standards in the RDADeLE System . . . . .	90
4.2.4.1	Standards of Learning Objects (LOs) Content Distributions . . . . .	91
4.2.4.2	Metadata Standards . . . . .	92
4.3	The Computational Model of The RDADeLE System . . . . .	92
4.3.1	The Entities . . . . .	93
4.3.2	The Mechanisms . . . . .	96
4.4	RDADeLE Architecture Components . . . . .	98
4.4.1	Regional Grid Structure . . . . .	101
4.5	Contents and Services Management . . . . .	103
4.6	Fault-Tolerance with Replicated Information Service . . . . .	105
4.7	Review of existing e-learning architecture towards RDADeLE Requirements . . . . .	106
4.8	Summary . . . . .	107
<b>5</b>	<b>RDADeLE Agents' Specifications</b>	<b>108</b>
5.1	Introduction . . . . .	108
5.2	The RDADeLE Knowledge Representation . . . . .	109
5.3	RDADeLE Agents' Relationships . . . . .	109
5.3.1	Communication . . . . .	109
5.3.2	Agent Relationships . . . . .	110
5.4	RDADeLE Scenarios . . . . .	112
5.4.1	XRT Scenario . . . . .	112

5.4.2	RST Scenario . . . . .	113
5.5	MAS-based RDADeLE . . . . .	115
5.6	Functions and Descriptions of RDADeLE Agents . . . . .	119
5.6.1	Administrative Agents . . . . .	120
5.6.1.1	Administrative Agent Architecture . . . . .	122
5.6.1.2	Administrative Agent Design Model) . . . . .	123
5.6.1.3	Administrative Agent Algorithms . . . . .	124
5.6.2	Regional Agents . . . . .	125
5.6.2.1	Regional Agent Architecture . . . . .	127
5.6.2.2	Regional Agent Design Model . . . . .	129
5.6.2.3	Regional Agent Algorithms . . . . .	130
5.6.3	Node Agents . . . . .	131
5.6.3.1	Node Agent Architecture . . . . .	132
5.6.3.2	Node Agent Design Model . . . . .	133
5.6.4	Service Agents . . . . .	135
5.6.4.1	Service Agent Architecture . . . . .	135
5.6.4.2	Service Agent Design Model . . . . .	137
5.6.5	Learner Agents . . . . .	137
5.6.5.1	Learner Agent Architecture . . . . .	138
5.6.5.2	Learner Agent Design Model . . . . .	139
5.7	Review of RDADeLE agents' specifications towards RDADeLE Re- quirements . . . . .	140
5.8	Summary . . . . .	141
<b>6</b>	<b>RDADeLE Implementation</b>	<b>142</b>

6.1	Introduction . . . . .	142
6.2	RDADeLE System Configuration . . . . .	143
6.2.1	JADE Platform Configuration . . . . .	145
6.2.2	Configuration of Learning Objects' information and Grid Services	146
6.2.2.1	The XML-based Registries Technique (XRT) . . . . .	146
6.2.2.2	The Registered-based Services Technique (RST) . . . . .	148
6.2.3	Grid Environment Configuration . . . . .	150
6.2.4	Node Configuration . . . . .	151
6.2.5	Information Service . . . . .	152
6.2.6	Regional Policy Configuration . . . . .	152
6.3	Simulation Validation . . . . .	153
6.3.1	Verification of Grid Configuration . . . . .	153
6.3.2	Verification of LOs' Information and Grid Service Configuration	156
6.3.3	Verification of Searching for Learning Objects' Information and Grid Services . . . . .	158
6.3.4	Verification of Regional Policy Application . . . . .	161
6.4	Review of RDADeLE implementation towards RDADeLE Requirements	163
6.5	Summary . . . . .	164
<b>7</b>	<b>Results and Evaluation</b>	<b>165</b>
7.1	Introduction . . . . .	165
7.2	Deployment Environment Setup . . . . .	166
7.3	Scalability . . . . .	166
7.3.1	Number of Regional Grids (Non-Main Containers) . . . . .	167
7.3.2	Number of Grid Services Agents . . . . .	173

7.3.3	Discussion . . . . .	174
7.4	Reliability . . . . .	176
7.4.1	Discussion . . . . .	181
7.5	Efficiency in Searching for Learning Objects' Information and Grid Services . . . . .	182
7.5.1	Searching for Learning Objects' Information . . . . .	182
7.5.2	Searching for Registered-based Grid Services . . . . .	183
7.5.3	Discussion . . . . .	186
7.6	Review of RDADeLE implementation result towards RDADeLE Re- quirements . . . . .	186
7.7	Summary . . . . .	187
<b>8</b>	<b>Conclusion</b>	<b>188</b>
8.1	Summary . . . . .	188
8.2	Contributions . . . . .	189
8.3	Future Work . . . . .	192
	<b>Bibliography</b>	<b>194</b>
<b>A</b>	<b>Java Code of RDADeLE implementation</b>	<b>207</b>

## List of Tables

2.1	Properties of the Main Grid Simulators . . . . .	36
2.2	Comparison of Some Main Grid Simulators . . . . .	36
3.1	Main Agent Architectures . . . . .	46
3.2	OOP versus AOP . . . . .	78
3.3	Main Characteristics of Major Agent Platforms . . . . .	78
3.4	Comparisons of Some Agent Platforms . . . . .	79
4.1	Administrative Regions of the Kingdom of Saudi Arabia . . . . .	85
4.2	Distribution of Universities, Colleges and Institutions Among Admin- istrative Regions of the Kingdom of Saudi Arabia . . . . .	86
4.3	World and the Kingdom of Saudi Arabia Population (thousands) . . .	87
6.1	Components of Regional Grids . . . . .	154
7.1	Scalability of Number of Regional Grids Using the XRT Technique . .	170
7.2	Scalability of Number of Regional Grids Using the RST Technique . .	172
7.3	Scalability of Number of Grid Services Agents . . . . .	174
7.4	Mean Search Time for Different Numbers of Registries Using the XRT Technique . . . . .	182
7.5	Mean Search Time for Different Registered Services Using the RST Technique . . . . .	184

## List of Figures

2.1	How Grid Works . . . . .	12
2.2	Layer of Grid Architecture . . . . .	20
2.3	Web Services Architecture . . . . .	25
2.4	Open Grid Services Architecture (OGSA) . . . . .	26
2.5	Open Grid Services Infrastructure (OGSI) . . . . .	27
3.1	KQML Structure . . . . .	49
3.2	UML JADE Architecture . . . . .	74
4.1	Administrative Regions in the Kingdom of Saudi Arabia . . . . .	86
4.2	RDADeLE System Deployment in the Administrative Regions in the Kingdom of Saudi Arabia . . . . .	89
4.3	RDADeLE Overview . . . . .	99
4.4	Architecture of Regional Grid Node of RDADeLE . . . . .	102
4.5	Contents and Services Management . . . . .	104
4.6	Fault-Tolerance With Replicated Information Service . . . . .	106
5.1	Class Diagram Association . . . . .	111
5.2	XRT Scenario in RDADeLE . . . . .	114
5.3	RST Scenario in RDADeLE . . . . .	116
5.4	Hierarchical Agent Organisation of MAS-based RDADeLE . . . . .	118
5.5	Administrative Agent Architecture . . . . .	123



5.6	Registration of Regional Grid . . . . .	124
5.7	Listing of Registered Regional Grids . . . . .	124
5.8	Regional Agent Architecture . . . . .	128
5.9	Search Request for LOs' Information . . . . .	129
5.10	Registration of Node . . . . .	129
5.11	Node Agent Architecture . . . . .	134
5.12	Registration of Learner . . . . .	134
5.13	Service Agent Architecture . . . . .	136
5.14	Registration of Grid Service . . . . .	137
5.15	Learner Agent Architecture . . . . .	139
5.16	Search Request for Grid Services . . . . .	140
6.1	Agent Types . . . . .	145
6.2	JADE Platform . . . . .	146
6.3	The XML-based Registries Technique (XRT) . . . . .	147
6.4	DCMS Meta-data Example . . . . .	148
6.5	The Registered-based Services Technique (RST) . . . . .	149
6.6	Regional Grid Configuration . . . . .	151
6.7	Node Properties . . . . .	151
6.8	Regional Policy . . . . .	153
6.9	Regional Grid Verification . . . . .	155
6.10	All Regional Grids (Containers) . . . . .	155

6.11	Components of Arab League . . . . .	155
6.12	Components of European Union . . . . .	156
6.13	Components of Indian Subcontinent . . . . .	157
6.14	Registry Content List . . . . .	158
6.15	Java Code for Building and Registering Grid Service . . . . .	159
6.16	Verification of XML Search in One Registry . . . . .	160
6.17	Verification of XML Search in Three Registries . . . . .	160
6.18	Verification of Registered Services Search . . . . .	161
6.19	Applying Regional Policy in the XRT Technique . . . . .	162
6.20	Applying Regional Policy in the RST Technique . . . . .	163
7.1	The Case Study Data Using RST Technique . . . . .	168
7.2	Extension of Riyadh Regional Grid . . . . .	168
7.3	Regional agent “Riyadh0@RDADeLE” Serving Sub-regional Grid “Riyadh0”	169
7.4	Regional agent “Riyadh1@RDADeLE” Serving Sub-regional Grid “Riyadh1”	170
7.5	Regional agent “Riyadh2@RDADeLE” Serving Sub-regional Grid “Riyadh2”	170
7.6	Number of Regional Grids Against Memory Consumption Using the XRT Technique . . . . .	171
7.7	Error of Maximum Number of Regional Grids Using the XRT Technique	171
7.8	Number of Regional Grids Against Memory Consumption Using the RST Technique . . . . .	172
7.9	Error of Maximum Number of Regional Grids Using the RST Technique	172

## LIST OF FIGURES

---

7.10 Comparison of Regional Grid Scalability using the XRT and the RST Techniques . . . . .	173
7.11 Error of Maximum Number of Registered Grid Services in a Regional Grid . . . . .	174
7.12 The Command Line to Store DF Catalogue into a Database . . . . .	175
7.13 Fault Tolerance Without Replicated Main Containers . . . . .	178
7.14 Fault Tolerance with Replicated Main Containers . . . . .	179
7.15 Activating Replication Services on Master Main Container . . . . .	180
7.16 Generating Replication of Main-Container Producing Main-Container-1	180
7.17 Generating Replication of Main-Container Producing Main-Container-2	181
7.18 Number of Registries Against Mean Search Time Using the XRT Technique . . . . .	183
7.19 Number of Registered Services Against Mean Search Time Using the RST Technique . . . . .	185
7.20 Mean Search Time Comparison . . . . .	185

# Chapter 1

## Introduction

e-Learning as a complex system is becoming influential in higher education today and is increasingly used in college and university courses. Although the e-learning system is a dynamic environment, it is still searching for a permanent, comfortable and serviceable position that is to be controlled, monitored, managed, flexible, accessible and continually up-to-date with the wider university structure. Once synonymous with distance learning, e-learning has quickly evolved to include not only courses that are taught primarily online and over a distance, but also to include traditional courses that have been enhanced with electronic elements which are called hybrid courses. According to the Educause Center for Applied Research (ECAR) respondent summary, 70% of all U.S. institutions offer distance learning and 80% of U.S. institutions offer hybrid courses [109].

### 1.1 Motivation and Significance

With the emergence of the Internet as the backbone of global communication and information exchange, greater attention has been paid to controlling, managing and monitoring complex systems and system of systems<sup>1</sup>. e-Learning systems are one example of complex and large scale systems. An abundance of information is currently presented to learners, employees and the general public by either applying conventional or advanced (electronic) methods. Many academic institutions and training centres worldwide are embracing web-delivered instruction. In recent years, new pub-

---

<sup>1</sup>Modern systems that comprise system of systems problems are not monolithic, rather they have five common characteristics: operational independence of the individual systems, managerial independence of the systems, geographical distribution, emergent behavior and evolutionary development [129].

## 1.1 Motivation and Significance

---

lic and private universities have been established to offer degree programs delivered exclusively online.

Involvement of data grid and agent technology as support for e-learning is beneficial in advancing controlling, managing and monitoring this paradigm [104] [92]. Data grid is one of the most interesting fields about the interaction of data within the grid environment. Data grid is a grid computing system that deals with large amounts of controlled, shared and managed distributed data. An intelligent agent is a software agent that exhibits some form of artificial intelligence that assists users and acts on their behalf in performing repetitive computer-related tasks.

The research aim is to integrate data grid technology with agent technology in the education and business fields by producing the managed and monitored Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE) system. This paradigm exploits facilities of the grid and the intelligent agent to search and analyse information collected from all over the grid environment.

While our work is applied generally we have taken the Kingdom of Saudi Arabia (KSA) as our case study. There are many reasons to adopt and embrace the RDADeLE system. These reasons can be seen in the following areas:

1. Demographics: an increase in the number of students requires opening of more universities and colleges which in turn require more faculty staff. For instance, the KSA has 252 government and private universities and colleges that grant bachelor, master and doctorate degrees in different fields. The number of registered students in 2009 in universities and colleges, under the umbrella of the Higher Education (HE) and the Technical and Vocational Training Corporation (TVTC), was more than 666662 students, and the number of faculty staff was more than 27964 members [28]. These statistics of the education system in the KSA provide us an indication of growing number of students, faculty

## 1.2 Problem Statement

---

staff, universities, colleges and institutions which encourage us to build a robust managed, monitored and dynamic system.

2. Economics: Building a robust and dynamic system which helps in organising and controlling institutions, their activities and their components will allow for a more cost effective accommodation of students in institutions.
3. Information: A large amount of information is available on the web; this should be exploited by organising and sharing it between different environments of universities and colleges.

## 1.2 Problem Statement

We will list the existing problems and issues which will be resolved by our e-learning model and architecture which is based on both agent and grid technologies. The learning process is evolving rapidly and the tools which are used in the learning process are evolving as well. Managing and monitoring such a system of systems (e.g. e-learning system) is not easy. The difficulty of managing and monitoring system of systems comes from managing and monitoring an increasing number of their components and elements. Besides that, to manage and monitor system of systems it requires interoperability between technologies which are used in such systems.

In the World Wide Web, there are many environments which are adopted to facilitate learning services over the internet. These web services use different kinds of technology and language to present services to learners. Architectures of these environments differ from each other depending on the needs of learners in each environment. Since learners interact with a large amount of information over the internet and a large number of components, focus in this research will be on managing and monitoring the components and the activities in the e-learning systems as a complex

## 1.2 Problem Statement

---

system. This leads us to integrate both the data grid environment with software agents in the e-learning environment context.

We cite in particular the most important issues of managing and monitoring complex and large systems without attributing any specific meaning to the order in which they are listed:

- Flexible grid-based learning architecture: Most of the conceptual architectures of grid-based learning environments concentrate on computational grid [34]. Researchers have paid a little attention to data grids when applying techniques in order to handle the LOs.
- Flexible and efficient agent-based learning architecture: Agents can provide a useful abstraction in the e-learning environment as well as provide services that are dynamic and robust. Using features of the agent is strongly recommended in the grid-based e-learning environment.
- Relationship to diagrammatical Unified Modeling Language (UML): As UML has gained wide acceptance, we argue that it is helpful to allow translation UML diagrams in any proposal of formal framework of grid/web and agent services.
- Integration of heterogeneous systems' capabilities in data distribution context: The major challenge which researchers are facing in grid technology is how different systems could be integrated to each other in order to accomplish a grid service among them.
- There is a general consensus that pedagogy and learning methodologies are now inextricably linked with technology. The nature of these interactions, however, remains little understood and there is a need for much more research into the impact of online learning techniques on the actual learning experience. The

## 1.2 Problem Statement

---

various responses to the Task Force survey indicated a growing acceptance, that technology can be used successfully to enhance learning irrespective of the mode in which learning takes place [109].

Our investigations will focus on adopting and adapting appropriate approaches of integrating data grids with agent technology in an e-learning context and deploying them as a framework for specifying and validating distributed data grid services so the framework can be managed and monitored. After this phase, we will focus on establishing rules and policies for each regional grid as well as for the global grid within an e-learning environment. Finally, the reflective model has to be enriched with management and monitoring capabilities using agent technology to manage and monitor the e-learning system. All phases are validated with implementation guided by case studies. In some detail, the conceptual framework we are endeavouring to develop in this thesis enjoins the following characteristics:

- Integrating data grid technology with intelligent agent technology within an e-learning context in order to provide grid service.
- Deriving conceptual model behaviours from UML activity and sequence diagrams which depict activity flows.
- Developing policies and rules which govern the regional grids' environment as well as global grids.
- Building infrastructure which manages and monitors the e-learning environment.

### 1.2.1 Research Question

Based on what we have presented earlier the research question is:



## 1.2 Problem Statement

---

**How can we produce a dynamic, extensible and flexible architecture which is capable of managing and monitoring the e-learning environment as a system of systems?**

This question gives rise to further questions. These are:

- **Managing**

- How can agents work dynamically within systems of systems and e-learning systems?
- How does the agent technology work to manage and monitor all the system's elements (e.g. data resources, nodes, and learners)?

- **Extensibility**

- How can the e-learning system be scalable?
- What is the role of grid technology in the e-learning system?

- **Flexibility**

- How does the agent technology facilitate ease of use in e-learning systems?
- What is the role of agent technology in searching for LOs' information and grid services?

- **Fault Tolerance**

- How can the e-learning system detect failures?
- How can the e-learning system work properly despite failures?

## 1.3 Research Contributions

This thesis presents three main novel contributions. The first is producing a dynamic architecture for the e-learning environment based on both grid and agent technologies which is called the RDADeLE system (See chapter 4). The RDADeLE system controls and monitors regional grids which are systems which could have subsystems comprising system of systems. The RDADeLE system controls all elements of these systems using agent technology which helps to accomplish processes of a dynamic and scalable environment such as the e-learning environment. The e-learning environment is dynamic and scalable since a large number of entities (e.g. nodes, resources, learners) connect and disconnect to it at anytime. Using agents in the RDADeLE system enriches it to be dynamic since agent technology has this ability. These agents can easily connect and disconnect from the RDADeLE system.

The second contribution is to produce a designing approach of data grid by dividing a whole grid into regional grids which are distributed (See chapter 4). The benefit of division is applicable for any dynamic and scalable environment. The division in the RDADeLE system is provided not only for division itself. It is provided to minimise problems of the whole project (i.e. distributed data grid) by determining and solving problems associated to each regional grid. Moreover, division is provided to build an autonomous regional grid especially in dynamic environments such as e-learning. The division has been adopted in RDADeLE in order to build a system of systems. Eventually, RDADeLE will combine these regional grids to build the whole system.

The third contribution is to produce a new agent-based grid simulator as a concept of designs and ideas prescribed in this thesis (See chapter 6). The RDADeLE system prototype simulates two techniques of composing and searching for LOs' in-

## 1.4 Research Methodology

---

formation and grid services. The first technique is **the XML-based Registries Technique (XRT)**. Learning objects' information in this technique are built using XML meta-data registries. The second technique is **the Registered-based Services Technique (RST)**. Grid services in this technique may have the LOs' information which can lead to learning objects' material. Moreover, grid services are built using agents that are registered so that they may be discovered.

## 1.4 Research Methodology

The following steps summarise the methodology used to achieve the goals of this research:

- Initially, an extensive study in grid and agent technology is employed, especially in the area of designing architectures.
- The primary goal of the extensive study is to identify and establish the concrete boundary of designing an appropriate architecture for an e-learning environment which is an agent-based data grid and consequently to determine the scope and the starting point of this research.
- An experimental evaluation is carried out based on the JADE platform to narrow down the research area and to identify deficiency in the existing architecture. This is achieved by analysis through simulation.
- Based on the starting point of this research, which is improving the performance of the e-learning environment, RDADeLE is developed and the result is our novel approach which is implemented and evaluated.
- A simulation is executed for the RDADeLE system with an analysis and comparison study of its performance.

## 1.5 Thesis Organisation

This thesis is organised into eight chapters including this chapter. The following is a brief description of the remaining chapters of this thesis.

**Chapter 2** outlines an overview of grid computing which reflects the literature review. This includes definitions of grid services, data grid, tools and languages used in grid services, relationships between grid services and web services, semantic grid, and some data grid applications.

**Chapter 3** outlines an overview of agent-based computing which reflects the literature review. This includes an overview of agent technology, agent design and development methodologies, multi-agent systems, and agent applications.

**Chapter 4** presents the RDADeLE system which is the new architecture for e-learning environments. This includes motivation, an architecture overview, components of the architecture and their functionalities. Also the regional grid is presented and described.

**Chapter 5** presents the agent specifications which includes specifications of all types of agents used in the RDADeLE system.

**Chapter 6** presents the implementation of the RDADeLE system. The implementation includes the prototype and simulation of the RDADeLE system.

**Chapter 7** presents the results of the research including evaluation. The results of the numerical implementation and testing of concepts of the RDADeLE system are presented at the end of the chapter.

**Chapter 8** presents the conclusions and the future work of the research.

# Chapter 2

## Grid Computing

### Objectives

---

- Review the definition, architecture, types and aims of grid computing.
  - Present some data grid applications.
  - Discuss the differences between web and grid services.
  - Review various types of grid service tools, languages and simulators.
- 

### 2.1 Introduction

Grid computing provides an environment where a widely distributed scientific and academic community shares its resources across different administrative and organisational domains. The purpose of grid computing is to facilitate large-scale computing, data-intensive computing, and to provide for collaboration in a wide variety of disciplines. Grid computing, therefore, enables the creation of a virtual environment which facilitates physical resources across different administrative domains in order to be beneficial. These resources are then abstracted into computing or storage units that can be transparently accessed and shared by large numbers of remote users.

The characteristics of grid computing is an attractive factor to be exploited in order to solve many problems in many fields. Those characteristics are not only CPU utilisation for high-performance computing environments but also about data man-

## 2.2 How Grid Computing Works

---

agement. Amazon and Google have built their own grid software to manage the growing number of transactions. Acxiom's environment grid had grown to 6,000 Linux nodes, processing more than 50 billion AbiliTec transactions per month (AbiliTec is a data-integration application)[144]. Many business and commercial entities, for instance British Broadcasting Corporation (BBC), transfer large files regularly, and work with grid technologies to make their data accessible at high speed across distributed networks. On data retrieval over different incompatible communication protocols some progress has been achieved in this field, especially in GridFTP [9]. This chapter presents an overview of grid computing covering the characteristics, applications, architectures, and grid tools.

## 2.2 How Grid Computing Works

Grids rely on middleware [7], which is advanced software/hardware that ensures seamless communication between distributed resources. Grids deploy powerful discovery services that discover unused resources across the grid in order to exploit them. Users have to verify their identities to access the grid through software interfaces running on their own computers. After authentication, the user will be able to describe the job to the grid resource broker, which is at the heart of the grid. The resource broker will find available resources that best fit the user's needs and job or application requirements by communicating with both the information service, to query information about software and hardware currently available, and the replica catalogue, to determine the location of required data. Once the application has selected the appropriate resources for the job, or has made advance reservations on selected resources, the job is submitted to those resources for execution. Finally, the resource broker sends the results back to the user as shown in Figure 2.1. All of these transitions are done transparently for the user, who perceives the grid as a single large and powerful

## 2.3 Characteristics of Grid Computing

---

computer [110][85].

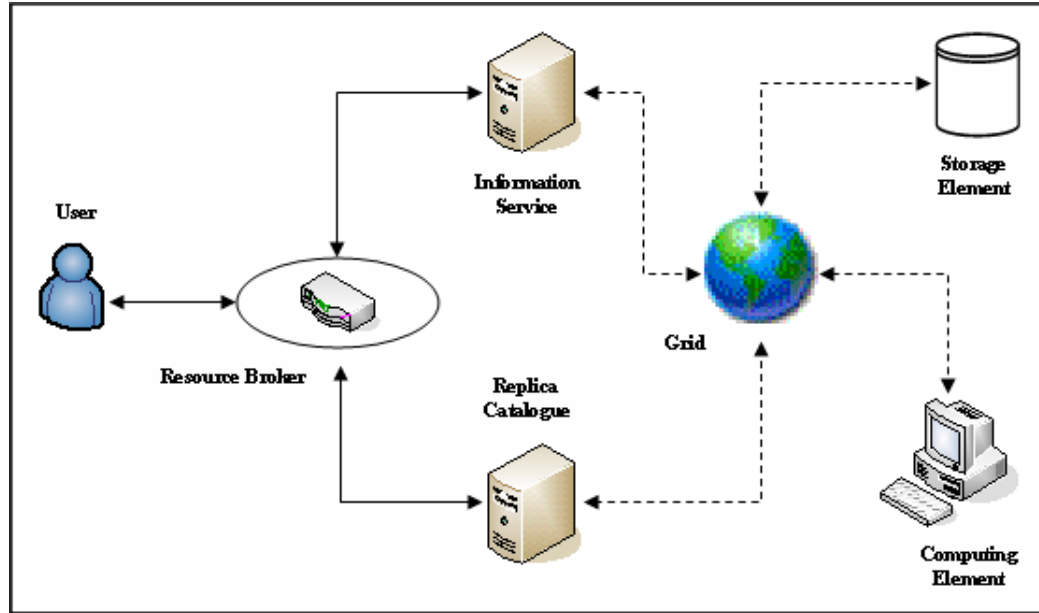


Figure 2.1: How Grid Works

## 2.3 Characteristics of Grid Computing

Grid computing aims to achieve the following goals [55] [15]:

- **Sharing of distributed and heterogeneous computing resources belonging to different organisations.**

Grid computing is the sharing, selection and aggregation of a group of resources such as supercomputers, mainframes, storage systems, data sources and management systems that operate in a network of computation [99] [3]. It promotes the sharing of distributed resources that may be heterogeneous in nature. The primary benefit of grid computing is the ability to coordinate and share distributed and heterogeneous resources such as Sharing Infrastructure and Resources in Europe (Sirene) [2]. Sirene is a cooperative association be-

## 2.3 Characteristics of Grid Computing

---

tween twelve European countries whose purpose is to share their infrastructure and resources.

- **Exploitation of underutilised resources**

In most organisations and companies there are large numbers of underutilised computing resources. Most of these resources are busy less than 5% of the time. Also, in some organisations these resources are relatively idle. Grid computing is designed to exploit these underutilised resources and increase the efficiency of resource usage. Users can also rent the resources that reside on the grid for executing their computationally intensive applications instead of purchasing their own dedicated (and expensive) resources.

- **Enablement and simplification of collaboration among different organisations**

Another capability of grid computing is the provision of an environment for collaboration between organisations. Grid computing enables very heterogeneous and distributed systems to work together, thereby simplifying collaboration between different organisations by providing direct access to computers, software and data storage.

- **Provision of a single login service with secure access to grid resources while protecting security for both users and remote sites**

Grids provide a single login service to all users over all distributed resources using grid authentication mechanisms. They also provide secure access to any information anywhere over any type of network. This is achieved by providing access control mechanisms which govern these resources.

- **Provision of resource management, information services, monitoring**



## 2.4 Grid Category

---

### **and secure data transportation**

The shared resources and networks involved in grid computing are difficult to manage and monitor, but grid computing is able to meet these challenges because of its architecture and protocols.

- **Designed to solve major problems**

Grids are designed to exploit underutilised resources, meaning that they can employ a large number of them to solve a major problem such as weather forecasting. These resources may be high capability devices such as high capacity disk storage and high performance computing.

- **Provision of quick results, delivered more efficiently**

The other attractive feature of grid computing is getting a result quickly and more efficiently, because it enables parallel processing; it may also have high capability devices. With grid computing, businesses can efficiently utilise computing and data resources and combine them for large capacity workloads.

## 2.4 Grid Category

Grid computing is becoming increasingly popular, with applications in many areas such as science and engineering. Grids have generally been categorised from the perspective of application as well as topology [16]. There are two wellknown types of grid: computational grid and data grid.

### 2.4.1 Computational Grid

A computational grid is a collection of computing resources that may represent computers on multiple networks and locations, heterogeneous platforms and separate

## 2.4 Grid Category

---

administrative domains with several owners. The purpose of a grid is to run very large applications. In this type of grid, most machines are high performance servers. The resources are aggregated so as to act as a unified processing resource. Computational grids emerged as a new paradigm for distributed computing. They promote the sharing of distributed resources that may be heterogeneous in nature and are created to solve complex engineering and scientific problems such as forecasting the weather and managing stock markets [55]. The increase in the speed and reliability of networks, in addition to the use of distribution protocols, is an important factor that has led to the use of grids, which enable users to remotely utilise resources owned by different providers. Computational grids fall into the categories of distributed supercomputing and high throughput computing depending on how they utilise resources. In the former, the grid executes the jobs in parallel on various resources, reducing completion times. Jobs requiring this category of grid are those that present large problems. By contrast, high throughputs increase job completion rates. The Northwest Indiana Computational Grid (NWICG)<sup>1</sup> is an example of a computational grid which offers three universities high speed and high bandwidth connection computational services and storage and data resources.

### 2.4.2 Data Grid

In this type of grid, a large amount of data is distributed and/or replicated to remote sites, potentially worldwide. In general, a data grid refers to a system responsible for storing data and providing access to parties authorised to share it. In other words, data grids provide an infrastructure for creating new information data repositories such as data warehouses or digital libraries that are distributed across several networks [85]. The aim of data grids is to overlap with heterogeneous distributed

---

<sup>1</sup>The Northwest Indiana Computational Grid, <http://dev.nwicgrid.org>.

## 2.4 Grid Category

---

database systems, which deal with various kinds of database management systems such as hardware, operating systems and network connections distributed across a heterogeneous environment. Data grid provides infrastructures to support data storage, discovery, handling, publication and manipulation. Enterprise data usually possess the characteristics of large scale, dynamic, autonomous, and distributed data sources. In the academic world, there is a desire to share expensive experimental data in order to coordinate research. In the business world, the need for data sharing is even more urgent, reasons include the requirement for business data to be in real-time for applications such as e-marketing, where it is very important to maintain an up-to-date and consistent product catalogue. The objective of data grids as presented in [43] is to integrate heterogeneous data archives into a distributed data management “grid”, in order to identify services for high performance, distributed, data-intensive computing, and to enable users to elicit relevant information from the distributed databases. Data grids are compatible with computational grids and can integrate storage and computation. One practical application of a data grid is the EU’s DataGrid<sup>2</sup>. This is a project financed by the European Union that relies upon developing computational grid technologies allowing distributed files, databases, computers, scientific tools and devices. The main goal of data grids is to build the next generation of computing infrastructure in order to develop and test the technological infrastructure that will enable the sharing of large-scale databases.

There is another category which is based on size. They are Cluster Grids, Enterprise Grids, Extraprise Grids, and Global Grids.

---

<sup>2</sup>**The DataGrid Project**, <http://eu-datagrid.web.cern.ch/eu-datagrid/>.

## 2.4 Grid Category

---

### 2.4.3 Cluster Grids

These are the smallest grids in size and scope. They are designed to solve problems for particular groups of people within the same department. They are implemented within campus intranets by incorporating PC's, data and servers to maximise the use of computer resources and to increase user job throughput. Cluster grids can therefore operate within a heterogeneous environment consisting of mixed server types, operating systems and workloads. Resources can be accessed at a particular point known within the grid which has only one job queue [58].

### 2.4.4 Enterprise Grids

Sometimes referred to as campus grids, these are collections of cluster grids; they enable inter-organisational cooperation. Enterprise grids allow departments to share resource collection under common policies without necessarily having to address the security and global policy management issues associated with global grids. For example, managers of IT departments in organisations might have dedicated computing resources. An enterprise grid would collect these resources from managers so that they would be shared by all concerned, even though they are owned by different people. This can be achieved through policies implemented within the environment. Sharing on this larger scale captures unused resources and makes them available to all authorised grid users.

### 2.4.5 Extraprise Grids

These connect two or more enterprise grids and, therefore, have several security domains. Every grid is autonomic and has its own access policies. Security management and resource management become both very difficult and more important in such an environment. Extraprise grids are set up between companies acting in partnership,

## 2.5 Grid Architecture

---

and between their networks and their customers. In this grid implementation, Virtual Private Networks (VPN) are used to make resources available. Some terms used by vendors to refer to Enterprise grids are [55]: ***Extra grids***, used by IBM to enable sharing of resources with external partners through VPNs. ***Partner grids***, used by platform computing to define grids between organisations with similar industries that need to collaborate on projects in order to reach a common goal.

### 2.4.6 Global Grids

This is a collection of enterprise and cluster grids connected by the Internet. Global grids offer a global view by virtualisation of distributed systems, especially in academia, where team members collaborate using systems that are geographically dispersed.

## 2.5 Grid Architecture

The traditional distributed technologies do not support an integrated approach to the wide variety of required services and resources, and they lack the flexibility and control needed to enable the type of resource sharing necessary. There is a need to define a grid software infrastructure to support the heterogeneous aspects of the grid. In [58] [56] [65], the grid strongly emphasises interoperability, as it is vital to ensure that virtual organisation participants can dynamically share heterogeneous resources. The grid infrastructure is based on a standard open architecture which facilitates extensibility, interoperability, portability and code sharing. This architecture organises components into layers, as shown in Figure 2.2 [58]. Components within each layer share common characteristics, but can build on the capabilities and behaviours of any lower layer.

- **Fabric Layer**

## 2.5 Grid Architecture

---

The fabric layer comprises the resources in the grid. This resource can be either a logical resources such as a distributed file system, computer cluster or distributed computer pool, or a physical resource such as computational resources, storage systems, catalogues, network resources and sensors.

This layer provides the lowest level of access to actual native resources and implements the low-level mechanisms that allow those resources to be accessed and used. More specifically, those mechanisms must include at least state enquiry and resource management mechanisms, each of which must be implemented for a large variety of local systems.

- **Connectivity layer**

The connectivity layer provides the core communication and authentication protocols required for grid-specific network transactions. These protocols provide cryptographically secure mechanisms for verifying the identified grid users and resources. Many communication protocols in the connectivity layer are drawn from TCP/IP protocols stack such as IP [123], ICMP [124], TCP[121], UDP [122] and DNS [105].

- **Resource Layer**

This layer builds on the connectivity layer to implement protocols that enable the use and sharing of individual resources such as the Grid Resource Access and Management protocol (GRAM) used to allocate and monitor resources. More specifically, two fundamental components of this layer are information protocols, for querying the state of a resource by calling fabric layer functions to control and access resources, and management protocols, used to negotiate access to a resource.

## 2.5 Grid Architecture

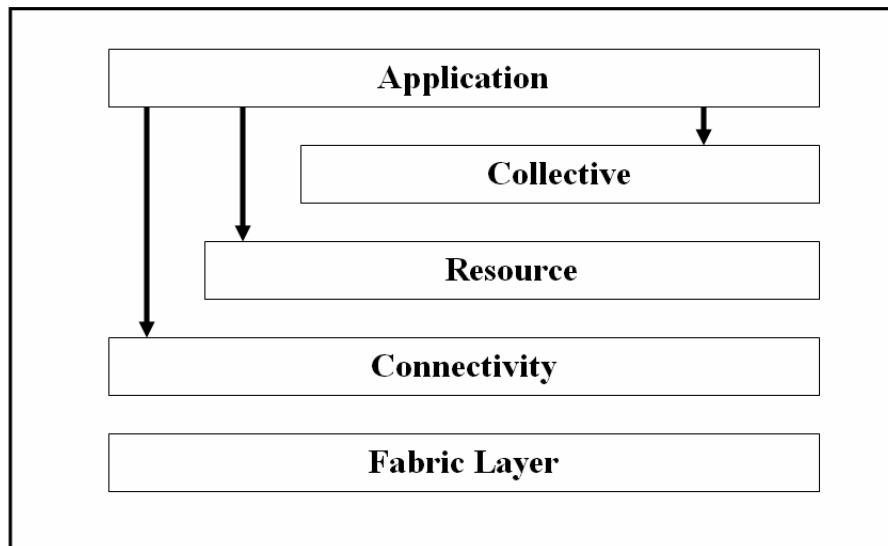
---

- **Collective Layer**

This layer provides protocols such as the Grid Resource Information Protocol (GRIP) for interacting across collections of resources. In other words, it focuses on the coordination of multiple resources. It includes directory, co-allocation, scheduling, brokerage, monitoring and diagnostics, data replication, software discovery, community accounting and payment services.

- **Application Layer**

This is the final layer in grid architecture, and contains the user applications that operate in a grid environment. It includes the languages and frameworks. These frameworks may themselves define protocols such as Simple Workflow Access Protocol (SWAP) [140], services, and/or an Application Program Interface (API).



**Figure 2.2:** Layer of Grid Architecture

## 2.6 Data Grid Applications

The possible applications to which data grids are being put are increasing to the extent that there can be said to be a definite trend towards this type of computing. Scientists and researchers use them for processing large amounts of data, while in the business world the number of grids is increasing exponentially. Most businesses have their own websites that detail their activities and offer services. Information, the central component in the field of education, is often transmitted electronically. In all these cases, the information stored remains no more than simply data distributed across different environments. Data grids provide the essential element of organisation, sharing and control. They comprise one of the most interesting fields by which data within a grid is manipulated. Some applications of data grids in various fields are as follows.

### 2.6.1 UK National Grid Service (NGS)

The National Grid Service (NGS)<sup>3</sup> provides UK researchers with coherent electronic access to computational and data-based resources for their research, regardless of location. The NGS is funded by the Joint Information Systems Committee (JISC), the Engineering and Physical Sciences Research Council (EPSRC) and the Council for the Central Laboratory of the Research Councils (CCLRC). The NGS was created in October 2003 and entered full production in September 2004. It is led and coordinated by the Science and Technology Facilities Council (STFC) in collaboration with the Universities of Manchester, Oxford and Edinburgh and the White Rose Grid at the University of Leeds. Researchers in different institutions across the UK can use its electronic infrastructure to obtain standardised access to computing and data resources and large scale facilities, as well as collaborating with colleagues from other

---

<sup>3</sup>**National Grid Service (NGS)**, <http://www.grid-support.ac.uk>.



## 2.6 Data Grid Applications

---

countries.

### 2.6.2 The Sakai Project

Sakai<sup>4</sup> is a free, open source online Collaboration and Learning Environment used to support teaching and learning, ad hoc group collaboration, support for portfolios and research collaboration. The community that uses Sakai often also provides the developers that create and improve it.

### 2.6.3 Enabling Grid for E-Science (EGEE) Project

The Enabling Grids for E-science (EGEE)<sup>5</sup> project integrates applications from many scientific fields to provide scientists from more than 90 institutions in 32 countries with a seamless grid infrastructure that operates 24 hours a day. EGEE's grid of more than 20,000 CPUs containing some 5 petabytes (5 million Gb) of storage provides the solution to the time and resource constraints of traditional IT infrastructures. It can run an average of 20,000 concurrent jobs. This capacity means that it is users' needs rather than system constraints that decide the use to which the grid is put. The grid provides large storage capacity, a wide range of bandwidth and computing power that answers all needs.

### 2.6.4 Sloan Digital Sky Survey (SDSS) Project

The most ambitious astronomical survey project ever undertaken, SDSS <sup>6</sup> is part of a change in the way science is conducted. It now processes vast amounts of data very quickly, so that it has become feasible for SDSS to map a quarter of the sky, producing detailed 3-D images of the more than 100 million celestial objects in that

---

<sup>4</sup>**Sakai project**, <http://sakaiproject.org/>.

<sup>5</sup>**The Enabling Grids for E-science (EGEE) project**, <http://www.eu-egee.org/>.

<sup>6</sup>**The Sloan Digital Sky Survey (SDSS)**, <http://www.sdss.org/>.

## 2.7 Web and Grid Services

---

segment, as well as determining their positions and brightness. Its range extends to the 10,000 known quasars at the edge of the visible universe.

### 2.6.5 TeraGrid

TeraGrid<sup>7</sup> is an open scientific discovery infrastructure combining leadership class resources at eleven partner sites to create an integrated, persistent computational resource coordinated through the University of Chicago's Grid Infrastructure Group working in partnership with Indiana University and several computing centres including the National Center for Supercomputing Applications, as well as the University of Chicago/Argonne National Laboratory and the National Center for Atmospheric Research.

## 2.7 Web and Grid Services

Emerging web services provide a framework for application-to-application interaction that grants access to business-to-business, e-science, and e-government services over the Internet. These services will allow a more extensive use of the web's functionality by supporting automated processes involving machine-to-machine cooperation and interaction.

There are so many service types which are provided to users in order to fulfil their needs. These services include organisation services, web services and grid services. The organisation services come from utilities which are provided by the organisation. On the other hand, web and grid services come from utilities which are provided by web and grid respectively. These services are provided to users after they go through some processes and procedures using some specific tools.

---

<sup>7</sup>**TeraGrid project**, <http://www.teragrid.org/>.

## 2.7 Web and Grid Services

---

A grid service is a web service that complies to a set of conventions that define how a client interacts with a grid service. These conventions, and other Open Grid Services Architecture (OGSA) mechanisms associated with grid service creation and discovery, provide for the controlled, fault resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications [128].

Web services and grid services are related strongly to each other. Grid services use the same standards as web services. However, there are more tools and standards used to create grid services. These tools are responsible for the main operations which include, for example, managing, monitoring, scheduling resources and services within the grid environment.

As defined by the World Wide Web Consortium [38], a web service is a software system identified by a Uniform Resource Identifier (URI) which is designed to support interoperable machine-to-machine interaction over a network. It has an interface that is capable of being described in a machine processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using Simple Object Access Protocol (SOAP) messages, typically conveyed using Hypertext Transfer Protocol (HTTP) with an Extensible Markup Language (XML) serialization in conjunction with other web-related standards. In [111] a web service is an interface that describes a collection of electronic operations that are network accessible through standardized XML messaging. It provides a set of functionalities to business and individuals and enables universal accesses to these functionalities. Web services has moved from a tightly coupled system to a loosely coupled system using existing web protocols, such as HTTP and Simple Mail Transfer Protocol (SMTP). Web services architecture consists of three entities:

- **Service Providers:** service providers create web services and publish these

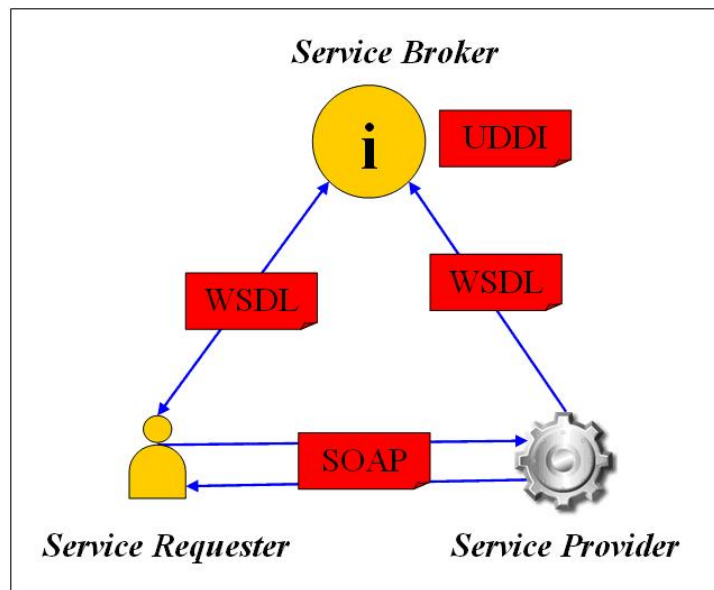
## 2.7 Web and Grid Services

---

services to the outside world by registering the services with service brokers using a registry.

- **Service brokers:** service brokers maintain a registry of published services.
- **Service Requesters:** Service requesters find required services by searching the service broker's registry. Requesters then bind their applications to the service provider to use particular services.

Figure 2.3 illustrates the interaction between service providers, service brokers, and service requesters in the publication, discovery, and consumption of web services [126].

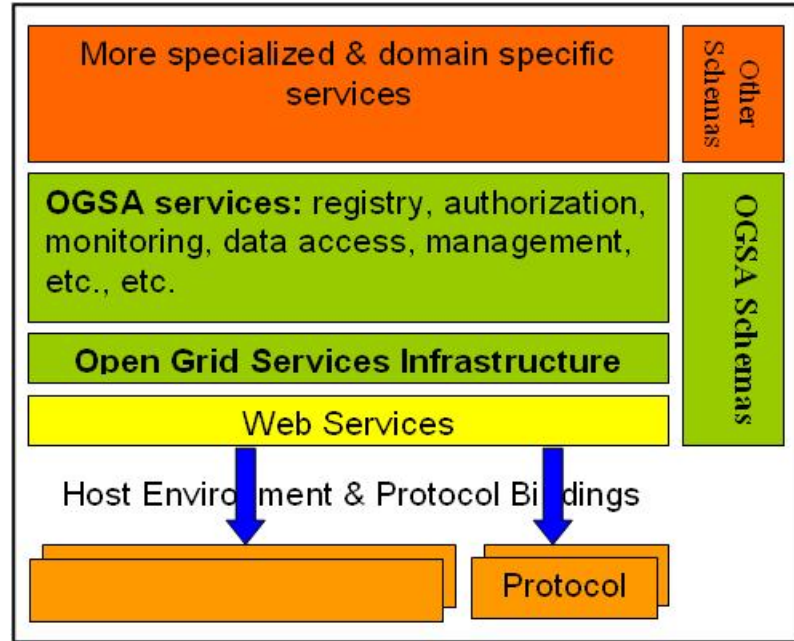


**Figure 2.3:** Web Services Architecture

The construction and creation of grid services depend on web services and other standards. The introduction of the Open Grid Services Architecture (OGSA) is evidence of the success of web services as a model for distributed computation, with the Open Grid Services Infrastructure (OGSI) as an interaction model for grid services [13].

## 2.7 Web and Grid Services

Figure 2.4 shows the OGSA Schemas which include OGSA services and Open Grid Service Infrastructure (OGSI) [54].



**Figure 2.4:** Open Grid Services Architecture (OGSA)

The OGSA has been embraced by The Open Grid Forum (OGF)<sup>8</sup> as the blueprint for standards-based grid computing. OGSA Work Group (OGSA-WG) defines the OGSA as follows: **Open:** is the process used to develop standards that achieve interoperability. **Grid:** is the integration, virtualization, and management of services and resources in a distributed, heterogeneous environment. **Services:** is delivering functionality as loosely coupled, interacting services aligned with industry-accepted web service standards. **Architecture:** the components, their organizations and interactions, and the design philosophy used.

The OGSA is a model to enable the integration of services and resources across distributed, heterogeneous, dynamic environments and communities. The OGSA was initiated by the Globus Project [54] and IBM to align grid technologies with Web

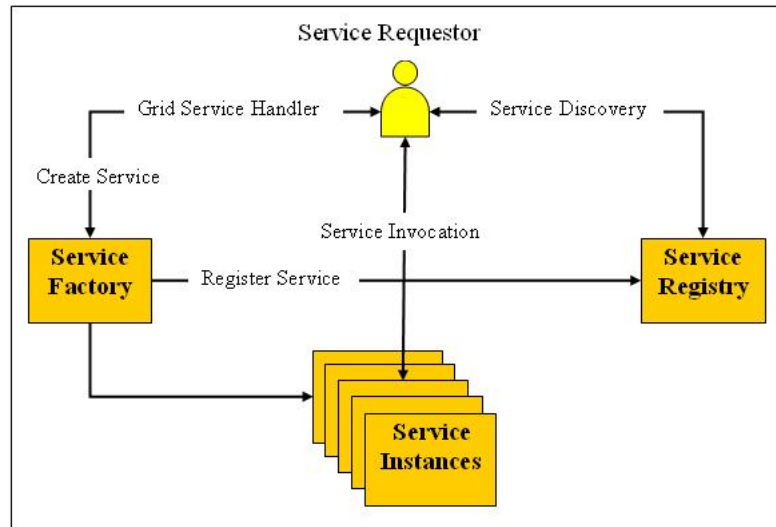
<sup>8</sup>The Open Grid Forum is an open community committed to driving the rapid evolution and adoption of applied distributed computing, <http://www.ogf.org/>.

## 2.8 Grid Service Tools

---

services technologies. The OGSA model adopts three web services standards [141] which are Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) and Web Services Inspection Language (WSInspection).

The OGSi defines mechanisms for creating, managing, and exchanging information among entities called grid services in order to build on both grid and web services technologies. The OGSi also introduces standard factory and registration interfaces for creating and discovering grid services [143]. The OGSi mechanisms associated with grid service creation and discovery provide for the controlled, fault-resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications. Figure 2.5 illustrates the role of OGSi to support grid services.



**Figure 2.5:** Open Grid Services Infrastructure (OGSI)

## 2.8 Grid Service Tools

The grid services tools are categorized into three parts. The first is grid middleware tools, the second is languages for grid services specification and composition, and the third is grid simulators.

## 2.8 Grid Service Tools

---

### 2.8.1 Middleware Tools

There is some existing open source grid projects which have been used as middleware within grid environments. These include:

#### 2.8.1.1 Globus Toolkit

The open source Globus<sup>9</sup> toolkit is a fundamental enabling technology for grids. It combines the facility to securely share resources including computing power, databases and other tools with the preservation of local autonomy. It is at the centre of a half-billion dollar international science and engineering project industry, and forms the framework for leading IT companies' commercial grid products. The toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection and portability.

#### 2.8.1.2 JC-Grid

JC-Grid<sup>10</sup> is a framework that combines the computing power of several workstations, including PCs and Macs, to create ad hoc grids. It requires only the installation of Java Runtime and is not reliant on any OS system, making it practicable for any job requiring a large amount of CPU processing power. File transfers are done via the framework, reducing the need for file system sharing capabilities such as NFS. Workstations can be added or taken away at runtime without the job being interrupted.

---

<sup>9</sup>**The Globus Alliance** is a community of organizations and individuals developing fundamental technologies behind the Grid, <http://www.globus.org/>.

<sup>10</sup>**JC-Grid is a Java Grid Computing**, <http://jcgrid.sourceforge.net>.

## 2.8 Grid Service Tools

---

### 2.8.1.3 Grid-Gain

Grid-Gain<sup>11</sup> is a computational grid product that allows the execution of a piece of code to be parallelised onto a set of computing resources that can be of any kind, including laptops or desktops, workstations, rack-servers, mainframes or any other resource with a capacity greater than Java 4 and that is compatible with Java Virtual Machine (JVM). The resources do not have to be compatible with each other, and can usually be local, enterprise-based and/or global. Grid-Gain enables the power of grid computing to be used for applications that are much more basic and small-scale than highly complex ones such as oil exploration, financial risk computations and weather forecasting.

### 2.8.1.4 Jini-Middleware

Jini technology<sup>12</sup>, as a service-oriented architecture for a programming model which is based on Java technology, combines well-behaved network services and clients into secure distributed systems. The architecture can build the kind of adaptive, scalable, evolvable and flexible network system required in dynamic computing environments.

## 2.8.2 Languages for Grid Services Specification and Composition

As we have mentioned before grid services rely on web services in specifications and compositions. Thus grid services use most web services languages. However there are other languages used for grid services besides web services languages. These languages include Grid Service Markup Language (GSML ) and Grid Process Exe-

---

<sup>11</sup>**GridGain Technologies** started in Pleasanton, CA in 2005 by the group of people that realized that traditional grid computing solutions are still largely inadequate for the majority of businesses, <http://www.gridgain.com>.

<sup>12</sup>**The Community Resource for Jini technology**, <http://www.jini.org>.



## 2.8 Grid Service Tools

---

cution Language (GPEL). The following are protocols and languages which are used to describe and compose communicate grid services:

- **Grid Service Markup Language (GSML):** Grid Service Markup Language (GSML) [94] is an XML-based grid programming language, which allows end-users to program a grid on demand.
- **Grid Process Execution Language (GPEL):** Grid Process Execution Language (GPEL) [147] is a new grid description language which is proposed based on Business Process Execution Language for Web Services language BPEL4WS. GPEL is an XML-based language defined by a set of XML schemas.
- **Hypertext Transfer Protocol (HTTP):** Hypertext Transfer Protocol (HTTP) is a protocol for distributed, collaborative, hypermedia information systems [48]. Its use for retrieving hypertext documents, led to the establishment of the World Wide Web in 1990 by English physicist Tim Berners-Lee. There are two major versions, HTTP/1.0 and HTTP/1.1. HTTP/1.1 may be faster as it takes time to set up such connections.
- **Simple Object Access Protocol (SOAP):** SOAP is an XML-based message protocol [102], which is specified in a W3C specification. Moreover, the likely technology to provide a communication framework for transport XML-based messages anywhere across the net is SOAP which facilitates the communication between web services and their clients.
- **Web Services Description Language (WSDL):** WSDL 1.1 describes the messages going in and out of a service [33]. “Cut and Paste” mechanisms provide for interface composition, in the absence of the more advanced WSDL 2.0 constructs. The WSDL document uses the following elements in the definition

## 2.8 Grid Service Tools

---

of network services: Types, Message, Operation, Port Type, Binding, Port and Service.

- **Universal Description, Discovery and Integration (UDDI):** UDDI is a XML-based global [36], public or private online directory which enables business or individuals to list businesses that they provide as a service provider and to be discovered by other services around the globe. UDDI is an extensible data model that is designed to work with any service description language such as WSDL. Services in the registry have to be well classified according to their functionality in order to allow requesters to find services that satisfy their needs.
- **WS-Addressing:** Web Services Addressing (WS-Addressing) provides the mechanism for locating and accessing a service or a resource [66]. It consists of two components: a structure used to communicate a reference to a web service endpoint, and a set of message addressing properties, which associate addressing information with a particular message.
- **WS-Security:** Web Services Security (WS-Security) is a communications protocol providing a means for applying security to web services [50]. SOAP Message Security 1.0 specification provides enhancements to SOAP messaging, giving message integrity and confidentiality in end-to-end interactions.
- **Security Assertion Markup Language (SAML):** When partners or services need to communicate role or authorization attributes over the network [113], SAML is used to encode the assertions. Extensible Access Control Markup Language (XACML) is used to encode the assertions as well. X.509 Certificates are used as the token profile for SOAP message authentication.
- **Web Service Flow Language (WSFL):** Web Service Flow Language was

## 2.8 Grid Service Tools

---

created by IBM [93]. It is an XML-based language which provides the mechanism to deal with complex interactions between one or more services that act both as clients and servers, thus provides support for a business process.

- **Web Services Business Process Execution Language (WS-BPEL):** WS-BPEL is a modified version of Business Process Execution Language for Web services (BPEL4WS) [76], which has been recently defined to describe compositions of services based on the business process model. WS-BPEL specification builds on IBM's WSFL (Web Service Flow Language) and Microsoft's XLANG (Web service for Business Process design) which allows a mixture of the block structured and graph structured process models.

### 2.8.3 Grid Simulators

Many applications use simulation, including the examination of natural or human systems' functioning by modelling them, the improvement of technological performance, safety engineering, testing, training and education. Simulation can be used as a forecasting tool and as a means of discovering and exploring alternative environments and options. Of the few tools designed to simulate grid computing environments, the leading ones are MicroGrid, Gridsim and Simgrid.

#### 2.8.3.1 MicroGrid

This flexible tool allows researchers to experiment on heterogeneous physical resources. It supports grid applications using the Globus Grid middleware infrastructure[136]. Its basic architecture consists of the following elements:

- **Virtualization:** This provides a virtual grid environment by intercepting all direct uses of resources or information services made by the application, es-

## 2.8 Grid Service Tools

---

pecially mediating all operations that identify resources name, to use and/or recover information about them.

- Global coordination: The virtual time module in MicroGrid can determine the maximum rate at which simulation is practicable by calculating from the desired virtual and actual resources employed (both CPU capacity and bandwidth/latency) to achieve a balanced simulation across distributed resources.
- Resource simulation: Each of the individual resources comprising the simulation must also be simulated correctly and must provide real-time performance feedback. The simulation must proceed at the same rate as the virtual time employed by the simulation. While many resources may turn out to be vital, computing and communication provide the initial focus.

The MicroGrid simulator has been written in the C programming language. Although the simulation builds a virtual grid environment it is not an agent-based platform.

### 2.8.3.2 GridSim

The GridSim toolkit [27] comprehensively enables simulation of a heterogeneous assortment of resources, users, applications, resource brokers and schedulers. It can be used to simulate application schedulers for single or multiple administrative domain distributed computing systems such as clusters and Grids. Resource brokers (i.e. grid application schedulers), each of which is dedicated to an individual user, locate, select and collect a heterogeneous resource set in order to customise the results for that user's requirements and purposes. All users' jobs go through the central scheduler, which can flexibly adjust resource allocation in order to raise system utilisation or prioritise certain users. Among the many functions of GridSim are:

## 2.8 Grid Service Tools

---

- the modelling of heterogeneous types of resources.
- the modelling of resources operating under space- or time-shared mode.
- the definition of resource capability in the form of MIPS (Million Instructions Per Second) as per SPEC (Standard Performance Evaluation Corporation) benchmark.
- the location of resources in any time zone.
- the mapping of weekends and holidays depending on resources' local time in order to model non-Grid (i.e. local) workloads.
- advance booking of resources.
- simulation of applications with different parallel application models.
- the ability to perform heterogeneous, CPU or I/O intensive application tasks.
- a limitless number of application jobs that can be submitted to a resource.
- the ability to submit jobs from several users simultaneously to the same resource. This feature helps in building schedulers that can use different market-driven economic models for selecting services competitively.
- the specification of network speed between resources.
- the support of simulation of both static and dynamic schedulers.
- the statistical recording and analysis of all or selected operations using GridSim statistical analysis methods.

GridSim is written in Java. Although the SimGrid simulator builds virtual grid environments, it is not an agent-based platform.

## 2.8 Grid Service Tools

---

### 2.8.3.3 SimGrid

SimGrid [91] provides a set of core abstractions and functionalities that can be used to easily build simulators for specific application domains and/or computing environment topologies. This allows the simulation of arbitrary performance fluctuations and observe real resources due to background load. SimGrid implements the following core abstractions:

- **Agent:** An agent is an entity that makes scheduling decisions. It is defined by a code, private data and the location at which it executes jobs.
- **Location:** A location (or host) is the place in the simulated topology at which an agent runs. Thus it is defined by a computational resource, a number of mailboxes that enable communication with other agents, and private data that can be only accessed by agents at the same location.
- **Task:** A task is an activity of the simulated application. For the moment it can be a computation and/or a data transfer. A task is defined by the amount of computing power required, the data size, and data itself.
- **Path:** The low-level, original SimGrid layer in the software did not provide any abstraction for message routing among locations. This made the user's simulation of complex platforms much harder. SimGrid now provides a routing abstraction so that the user (and the scheduler) can rely on a logical platform topology rather than targeting the physical topology directly. A path is an agglomeration of communication resources representing a set of physical network links. Locations are then interconnected through paths. The simulated application cannot access links directly (likewise, that a real application does not choose which routers its packets go through).

## 2.8 Grid Service Tools

---

- **Channel:** Channel Communication between agents is embedded in the channel abstraction. A channel embodies the concept of communication ports opened by agents at locations.

SimGrid has been written in Java. SimGrid does use agents, but only for scheduling.

Table 2.1 shows some properties of some of the main grid simulators.

**Table 2.1:** Properties of the Main Grid Simulators

Tool	Organisation	Properties
MicroGrid	University of California at San Diego, U.S.A.	Runs emulations by executing actual application code on the virtual Globus Grid and thus requires more time to complete the application, emulates the Globus Grid environment for resource management, <a href="http://www-csag.ucsd.edu/projects/grid/">http://www-csag.ucsd.edu/projects/grid/</a>
GridSim	University of Melbourne, Australia	Supports simulation of space-based and time-based, large-scale resources in the grid environment, simulates economy-based resource scheduling systems in grids, <a href="http://www.gridbus.org/gridsim/">http://www.gridbus.org/gridsim/</a>
SimGrid	University of California at San Diego, U.S.A.	Simulates single or multiple scheduling entities and timeshared systems operating in a grid computing environment, simulates distributed grid applications for resource scheduling, <a href="http://grail.sdsc.edu/projects/simgrid/">http://grail.sdsc.edu/projects/simgrid/</a>

Table 2.2 compares some of the main grid simulators.

**Table 2.2:** Comparison of Some Main Grid Simulators

Design	MicroGrid	GridSim	SimGrid
Simulated systems	Grid, resource scheduling systems	Grid, resource scheduling systems	Grid, resource scheduling systems
Usage	Emulator	Simulator	Simulator
Programming framework	Structured	Object-oriented	Object-oriented

## 2.9 Review of Data Grid Technology towards E-learning Requirements

Management is one of the major requirements for the e-learning architecture. Management is the ability to manage and share distributed data resources authorised under different domains. Data grid technology, the subject of the chapter, has been used successfully to manage system of systems and e-learning environments. E-learning environment, as system of system, need to manage storage resources. For example in [2] the heterogeneous distributed data resources are shared using grid middleware. Using grid middleware (OGSA-DAI), without alteration, will guarantee management and extensibility of data resource in the e-learning architecture. This means there is no enhancement of data grid is necessary.

The second requirement for the e-learning architecture is extensibility. Extensibility refers to the scalability of the e-learning architecture. A scalable architecture has the ability to accommodate increase number of data resources. Using grid middleware enriches the e-learning architecture to facilitate large-scale data-intensive computing and to provide for collaboration in a wide variety of disciplines. For example UK National Grid Service (NGS) and Enabling Grid for E-Science (EGEE) project, discussed in section 2.6, have used data grid technology to provide access to data resources and large scale facilities and provide large storage capacity. This grid middleware can also be used to ensure scalability in an e-learning architecture.

The remaining two requirements are flexibility and fault tolerance. These are not addressed by grid standards and implementations so a different solution and techniques are required. Research has revealed that agent technology has the properties required for these two requirements. Agent technology and how they address the requirements is the subject of chapter 3. The conclusion is that data grid is an ap-



## 2.10 Summary

---

propriate technology which will support the management and scalability requirement in an e-learning architecture.

## 2.10 Summary

This chapter surveyed the background to and related work on grid computing especially data grid. It described grid computing, including characteristics, architectures, and categories. This is followed by a survey of data grid applications, web and grid services, and grid tools and programming languages. Finally grid simulators are then surveyed, including McroGrid, GridSim and SimGrid.

# Chapter 3

## Agents-based Computing

### Objectives

---

- Review the definition, architecture, types of agent computing.
  - Present agent designing and development methodologies and languages.
  - Present characteristics and applications of multi-agent systems.
  - Discuss agent roles in grid computing.
  - Review agent platforms, simulators and comparisons.
- 

### 3.1 Introduction

Since the 1990s, research and development in agent-based systems has been an important and active area in information technology. The reason is that agent technology provides flexible, autonomous, dynamic, and distributed domains. The concept of an agent has been applied in many fields such as computer networks, software engineering, artificial intelligence, human-computer interaction, distributed and concurrent systems, mobile systems, computer-supported cooperative work, control systems, data mining, decision support, information retrieval and management, and electronic commerce [97]. This chapter presents an agent technology overview, agent design and development methodologies and languages, multi-agent systems, roles of agents

## 3.2 Agent Technology Overview

---

in grid computing, and agent platforms and simulators.

## 3.2 Agent Technology Overview

There is no agreement or universally accepted definition of the term *agent*. One definition specifies an agent is any entity that perceives its environment through sensors and acts on the environment based on its own reasoning capability; this definition includes human, robotic and software agents [127]. Agent technology has been exploited heavily on the web through applications in many fields including industry, business, education and training.

There are application domains where agent technologies play a crucial role. These applications include grid computing, where multi-agent system enable efficient use of the resources of high-performance computing infrastructure in science, engineering, medical and commercial applications; electronic business, where agent-based approaches support the automation and semi-automation of information gathering activities and purchase transactions over the Internet; the semantic web and grid, where agents provide services; bioinformatics and computational biology; monitoring and controlling other systems; resource management and military and manufacturing applications [97]. There are different impacts of agent technology in application domains which include assistance in designing of complex and distributes systems, a source technology in computing systems and a model for complex of real world systems. One of the most well-known agent applications [87] is video games, which have become a large part of many peoples' lives. The application of agent technology in video games has many aspects. One of the obvious benefits of video games is the elimination of risk to human life involved in any real-world application. They also make an excellent testbed for techniques in artificial intelligence. However, there are some general attributes and characteristics distinguish agents.

## 3.2 Agent Technology Overview

---

### 3.2.1 Characteristics of Agents

Agents have many features which make them suitable to be applied in many applications. The following are some of the main characteristics of agents [23]:

- **Autonomy:**

Agents are entities capable of committing effective operations in dynamic and open environments. Agents are used in environments in which they interact and cooperate with other agents. The autonomy and degree of autonomy has been defined by some paper authors which included in [100]. In multi-agent systems, agents can be distinguished from objects in that they are autonomous entities capable of exercising choice over their actions and interactions. Agents cannot, therefore, be directly invoked like objects. However, they may be constructed using object technology.

- **Adaptability and proactiveness:**

Adaptability is the change in behaviour so that it becomes suitable to a new situation. Proactiveness is the capability of taking the initiative, not driven solely by events. The agent can be designed to obtained adaptable architecture [42] in order to use this features in its operations.

- **Mobility:**

An agent could be roaming around and within a system on a specific route. This is the mobility which gives agents ability to migrate to a remote system, perform the tasks and then return the results. Mobile agent modeling can use ontology for interrelationships design [31]. The agent has been used heavily in the internet to support the internet applications on the internet framework [39].

## 3.2 Agent Technology Overview

---

- **Persistence:**

Persistent agents do not die after the task is completed they remain alive until the system terminates it. It can live on local or remote processing sites.

- **Goal Orientation:**

Each agent has been designed to fulfil a goal or aim. More advanced and complex agent types are dedicated for a specific goal for instance Belief-Desire-Intention (BDI) agents.

- **Communication, Collaboration, Cooperation:**

Agents have standards communication protocols in order to communicate with each other or to communicate with other components inside or outside its platform. Communicative features are important to enable agents to collaborate and cooperate in order to achieve goals.

- **Flexibility:**

Flexibility could be in the design, behaviour or in the manner in which the agent interacts. This feature allows agents to be flexible problem solvers.

### 3.2.2 Classifications of Software Agents

Agents are generally classified, whether the criteria be level of mobility (i.e. static or mobile), role, hybrid philosophy, exhibition of ideal and primary attributes such as autonomy, cooperation and learning, and the presence of a symbolic reasoning model such as deliberative or reactive. The following classification is based on the topology [114]:

- **Collaborative Agents:**

Cooperation between agents requires the development of a mutual understanding and sharing regarding the task in hand and the goal towards which they are

### 3.2 Agent Technology Overview

---

working. This kind of agent may have to negotiate in order to reach agreement on some matters. Filtering email is one application of collaborative agents [89].

- **Interface Agents**

Pattie Maes in [150], a key proponent of this class of agent, has defined it as “Computer programs that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular application. The metaphor is that of a personal assistant who is collaborating with the user in the same work environment”. The interface agent is used in the internet search engines [5] to support users in finding exactly the information consistent with his/her interest.

- **Mobile Agents**

Mobile agents are computational software processes capable of moving within a specific route or networks such as the WWW [114]. They have the ability to transport themselves from one machine to another. A mobile agent may interact with other agents and foreign hosts in order to gather information and return to its home after having performed the duties prescribed by the user. Mobile Agent can be used in many fields. But the security problem [131] of mobile agents is a barrier for applications.

- **Information/Internet Agents**

Information agents may be static or mobile [114], non-cooperative or social, and they may or may not have the capacity to learn. Hence, they have no standardised mode of operation. Information agents’ main function is to gather information, report what they retrieve from the WWW and return that information to where it came from. The last few years have witnessed a high rate of growth in revenue from single-item online auctions popularised by eBay and

## 3.2 Agent Technology Overview

---

similar websites [30] which are considered as applications of internet agents.

- **Reactive Agents**

This kind of agent responds in real time to changes in the environment. They do not possess internal, symbolic models of their environments. Brooks (1986) and Chapman (1987) first expounded their characteristics. Many theories, architectures and languages to describe this type of agent have since been developed. They are seen as simple, interacting with other agents in basic ways. Reactive agents has been used in many fields one of which is in [67] the manufacturing process control.

- **Hybrid Agents**

Hybrid (or layered) agents are a combination of two or more kinds of agent [114]. This is important since some applications need special kinds of agents in order to meet their requirements. Assets are amplified and deficiencies reduced by combining two types of agent to produce hybrids. Virtual environments [146] is an example of the hybrid agents applications.

- **Smart Agents**

Smart agents have the ability to learn and to gain experience as they react and/or interact with their external environment [114]. This kind of agent is both collaborative and autonomous. These features impart some degree of intelligence to these agents. Smart agent has been used in web knowledge mining [119] in the WWW field.

### 3.2.3 Main Agent Architectures

Maes defines an agent architecture as: [98] “A particular methodology for building agents. It specifies how the agent can be decomposed into the construction of a set

### 3.2 Agent Technology Overview

---

of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions and future internal state of the agent. An architecture encompasses techniques and algorithms that support this methodology.”

Kaelbling [82] considers an agent architecture to be “A specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the data and control flow among the modules. A more abstract view of an architecture is as a general methodology for designing particular modular decompositions for particular tasks.”

Fundamental objects of an agent architecture include:

- Agent: The Agent class defines the agent. An instance of this class exists for each agent executing a job on a given agent host.
- Agent Host: An agent host keeps track of all the agents executing jobs in the system. It will interact with other agent hosts to transfer an agent from one system to another.
- Agent Interface: An instance of this class envelopes an agent and provides access to it via a well-defined interface. It is also the primary conduit for communication between agents. An agent interface instance is the only agent which interacts with other agents executing on a given host.
- Agent Identity: An instance of this class uniquely identifies an agent. Agents use this information to identify the agents with whom they are interested in collaborating.

Table 3.1 presents attributes of main agent architectures [47]:



## 3.2 Agent Technology Overview

---

**Table 3.1:** Main Agent Architectures

Type of architecture	Approach	Subordination structure
Horizontal modular	Horizontal functional	Hierarchical
Blackboard	Functional	Hierarchical (Meta)
Subsumption	Vertical functional	Hierarchical
Competitive task	Vertical functional	Hierarchical (Competition)
Production rules	Functional	Hierarchical (Meta)
Classifier	Vertical functional	Hierarchical
Connectionist	Vertical functional	Egalitarian
Dynamic system	Vertical functional	Egalitarian
Multi-agent	Object/functional	Egalitarian

Three main agent architectures will be presented in more detail. They are reactive, cognitive and hybrid architectures.

### 3.2.3.1 Reactive Architecture

Reactive architecture does not include a central symbolic world model, and does not use complex symbolic reasoning. In 1985 Rodney Brooks, a researcher at MIT, outlined an architecture, called subsumption architecture, for building agents. Subsumption architecture is a hierarchy of task-accomplishing behaviours, each of which competes with others. Lower layers represent more primitive kinds of behaviour and have precedence over layers further up the hierarchy. Examples of this type of architecture are PENG1, situated automata, and Agent Network Architecture [150].

### 3.2.3.2 Cognitive Architecture

Cognitive architecture is mainly used for a broad, multiple-level, multiple-domain analysis of cognition and behavior which is essential structure and process of a broadly-scoped domain-generic computational cognitive model [112] [138]. A cognitive architecture provides a solid framework for more detailed modeling of cognitive phenomena, through specifying essential structures, divisions of modules, relations

## 3.2 Agent Technology Overview

---

between modules, and a variety of other aspects [137]. There are some famous cognitive architectures which include ACT-R , CLARION, CHREST, PRODIGY and SOAR.

### 3.2.3.3 Hybrid Architecture

In order to build a hybrid agent, it must be taken into account that architecture is composed of subsystems [149] which are arranged into a hierarchy of interacting layers. There are two examples of this kind of architecture, InteRRaP and Touring Machines. The control flow within layers may be horizontal or vertical layering. In horizontally layered architectures, the software layers are each directly connected to the sensory input and action output. In vertically layered architectures, sensory input and action output are each dealt with by, at most, one layer. The hybrid agent architecture in [139] includes both reactive and deliberative architectures.

### 3.2.4 Agent Communications

Agents have the ability to communicate with other agents, applications, and humans. The communication includes sending and receiving messages in order to achieve the goals of themselves or of the society/system in which they exist. Communication can enable the agents to coordinate their actions and behaviours to build more coherent systems. The degree of coordination is the extent to which they avoid unnecessary activities. Cooperation is coordination among non-antagonistic agents, while negotiation is coordination among competitive agents. Communication protocol is specified by a data structure with the following five fields [148]:

- Sender.
- Receiver(s).

## 3.2 Agent Technology Overview

---

- Language in the protocol.
- Encoding and decoding functions.
- Actions to be taken by receiver(s).

### 3.2.4.1 Speech Acts

Computational agents use spoken human communication as the model for communication. Speech act theory [14] [130] is the basis for analysing human communication. Speech act theory views humans' natural language as actions, such as requests, suggestions, commitments, and replies. A speech act [148] has three aspects which are Locution (the physical utterance by the speaker), Illocution (the intended meaning of the utterance by speaker), and Perlocution (the action that results from the locution).

Speech act theory uses the term performative to identify the illocutionary force of utterance. Performative verbs may include promise, report, convince, insist, tell, request, and demand. Speech act theory helps define the type of message by using the concept of the illocutionary force, which constrains the semantics of the communication act itself. The sender's intended communication act is clearly defined, and the receiver has no doubt as to the type of message sent.

### 3.2.4.2 Knowledge Query and Manipulation Language (KQML)

The knowledge query and manipulation language (KQML) is a protocol for exchanging information and knowledge. KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents [49]. All information for understanding the content of the message is included in the communication itself. Figure 3.1 [148] shows KQML structure.

## 3.2 Agent Technology Overview

---

```
(KQML-performative
      :sender      <word>
      :receive     <word>
      :language    <word>
      :ontology    <word>
      :content     <expression>
      ...)
```

**Figure 3.1:** KQML Structure

The terms *:content*, *:language*, and *:ontology* determine the semantics of the message. Other arguments, including *:sender*, *:receiver*, *:reply-with*, and *:in-reply-to*, are parameters of the message passing. The semantics of the message is defined by the fields *:content* (the message itself), *:language* (the language in which the message is expressed), and *:ontology* (the vocabulary of the “words” in the message). The language in a KQML is not restricted to one language. Knowledge Interchange Format (KIF) is used to express the messages. Other languages such as PROLOG, LISP, SQL can be used. The KQML performatives are organised into seven categories which include for example basic query performatives (such as *evaluate*, *ask-one*, *ask-all*), response performatives (such as *reply*, *sorry*), and networking performatives (such as *register*, *unregister*, *forward*, *broadcast*).

### 3.2.4.3 Knowledge Interchange Format (KIF)

The ARPA Knowledge Sharing Effort has produced the Knowledge Interchange Format (KIF) [63] logic language for describing the information content transmitted and the conceptual vocabularies. The KIF is a particular logic language which has been proposed as a standard to describe things within some systems such as experts systems, databases, and intelligent agents. KIF language has some advantages which include understandability and translation. Understandability means that it is readable by both computer systems and people. Translation means the KIF language

## 3.2 Agent Technology Overview

---

could be a means or mediator between other expression languages. KIF can be used to describe procedures to write programs for agents to follow in order to achieve desired aims. The KIF has been used in composition of web services [60].

### 3.2.4.4 FIPA-Agent Communication Language (FIPA-ACL)

Foundation for Intelligent Physical Agents (FIPA) ACL is based on speech act theory. KQML and FIPA ACL messages look syntactically identical.

A FIPA ACL message contains a set of one or more message parameters which include for example *sender*, *receiver*, *performative* and *content* which is expressed in a content language, such as FIPA-SL or FIPA-KIF [52]. The only parameter that is mandatory in all ACL messages is the performative (i.e. the type of the communicative act of the ACL message) which includes for example *Inform*, *Request*, *Agree*, *Refuse* and *Failure*.

### 3.2.4.5 Ontologies

Using the same language in communication between agents is not sufficient for understanding between them. Agents need to use same terminology [46]. Ontology is defined as a specification of the objects, concepts, and relationships in an area of interest. The ontology must describe the relationship. The classes and relationships must be represented in the ontology; but the instances of classes need not be represented. The agent's developer must use a specific ontology to represent the agent's knowledge. All agents that share the same ontology for knowledge representation have an understanding of the word in the agent communication language.

## 3.2 Agent Technology Overview

---

### 3.2.5 Agent Interaction Protocols

Interaction protocols administrate the exchange of a series of messages among agents (conversation). The objective of the protocols is to maximize the utilities of the agents [125]. The following are the main interaction protocols:

#### 3.2.5.1 Coordination Protocols

In a multi-agent environment coordination between agents is recommended for many reasons [148]. These reasons include to satisfy group goals, work properly with limited resources, dependency between agents' actions, and to complete system activities. It appears that coordination is important in an environment which distributes both control and data. Agent Coordination Context (ACC) is used to support interaction between an individual agent and the MAS as a whole [107].

#### 3.2.5.2 Cooperation Protocols

Cooperation protocols are used to decompose and then distribute tasks [148]. This approach can reduce the complexity of tasks since smaller subtasks require less capable agents and fewer resources. The system designer programs the system to include task decomposition during implementation. The following mechanisms are some commonly used to distribute tasks: Contract Net, Market Mechanism and Blackboard. Prometheus methodology - a methodology for agent-oriented software engineering - has been extended to support agent cooperation [8].

#### 3.2.5.3 Contract Net

The contract net protocol is an interaction protocol for cooperative problem solving among agents [148]. It is based on finding an appropriate agent to work on a given task. The *manager* agent announces a task that needs to be performed, receives and

## 3.2 Agent Technology Overview

---

evaluates bids from *contractor* agents, and awards a contract to a suitable contractor. The *contractor* agent receives task announcements, evaluates its capabilities to respond, responds, and performs the task if its bid is accepted. Many applications have adopted contract net protocol in their systems, for example in Manufacturing Systems [72].

### 3.2.5.4 Market Mechanisms

Market mechanisms are needed when there are a large number of unknown agents [148]. Computational economies approach is based on market mechanisms and are effective for coordinating the activities of many agents with minimal direct communication between agents. There are two types of agents, *consumers* and *producers*. *Consumers* exchange goods and *producers* transform some goods into other goods. Some manufacture systems have adopted Market Mechanism in their systems [106].

### 3.2.5.5 Blackboard Systems

The idea of a Blackboard system is based on the concept of a blackboard as a workspace for agents (called Knowledge Source KS) for developing a solution for a problem [148]. Agents participate cooperatively to solve a problem by posting its expertise on the blackboard. The problem is solved when there is sufficient information to make a contribution. The following are some important characteristics of blackboard systems: independence of expertise, diversity in problem-solving techniques, flexible representation of blackboard information, common interaction language, event-based activation, need for control, and incremental solution generation. There are some applications of the Blackboard system in the e-learning environment [96] especially in the investigations of issues related to the e-learning environment.

### 3.3 Agent Design and Development Methodologies and Languages

---

#### 3.2.5.6 Negotiation

Negotiation is a process between two or more agents to reach a joint decision, each agent trying to reach an individual objective [148]. Negotiation has the following major features: the language used by participating agents, the protocol followed by the agents as they negotiate, the decision process that each agent uses to determine its positions, concessions, and criteria for agreement. Environment-centered and agent-centered approaches are two techniques of negotiation systems. Attributes of negotiation mechanism include efficiency, stability, simplicity, distribution, and symmetry. There are many negotiation applications in the agent field, one example is that negotiation strategy is used in automated processes [135].

### 3.3 Agent Design and Development Methodologies and Languages

Agents and multi-agent systems are relatively new area. There is no standard definition of agent and multi-agent systems. Hence, there is no specific methodology for designing and the development of agent and multi-agent systems. The following are some current agent-based software engineering methodologies [46]. But before that we will present some agent related terminologies.

#### 3.3.1 Terminology

Being a relatively new research field, agent-based software engineering currently has a set of closely related terms used in research papers, we will thus try to clarify and explain the terms and their relations below. Agent-Oriented Programming (AOP) is seen as an improvement and extension of Object-Oriented Programming (OOP). Since the word “Programming” is attached, it means that both concepts are close to the programming language and implementation level. The term “Agent-Oriented



### 3.3 Agent Design and Development Methodologies and Languages

---

Programming” was introduced by Shoham in 1993 [145]. Agent-Oriented Development (AOD) is an extension of Object-Oriented Development (OOD). The word “Development” is sometimes interpreted as “Programming”, on the other hand it is frequently interpreted to include the full development process that covers the requirement specification and design, in addition to the programming itself. Software Engineering with Agents, Agent-Based Software Engineering, Multi-agent Systems Engineering (MaSE) and Agent-Oriented Software Engineering (AOSE) are semantically equivalent terms, but MaSE refers to a particular methodology and AOSE seems to be the most widely used term. The difference between AOSE and AOD, is that AOSE also covers issues such as re-use and maintenance of the agent-system in addition to the development of the system itself. However, to be on the safe side, one should omit the use of the term AOD since it can easily be misinterpreted, as pointed out earlier (due to the different interpretations). The term Agent-based Computing can be applied to describe all issues related to agent-oriented software engineering, but it also covers issues regarding how and what agents compute.

#### 3.3.2 Agent-oriented Methodologies

Agent-oriented methodologies include Gaia, Prometheus, Tropos, MaSE, and ROADMAP.

##### 3.3.2.1 Gaia

The Gaia methodology for agent-oriented analysis and design has been presented by Wooldridge, Jennings and Kinny [151] [152]. Gaia is a general methodology that supports both the micro-level (agent structure) and macro-level (agent society and organisation structure) of agent development, it is however no “silver bullet” approach since it requires that inter-agent relationships (organisation) and agent abilities are static at run-time. The motivation behind Gaia is that existing methodologies fail

### 3.3 Agent Design and Development Methodologies and Languages

---

to represent the autonomous and problem-solving nature of agents; they also fail to model agents' ways of performing interactions and creating organisations. Using Gaia, software designers can systematically develop an implementation-ready design based on system requirements. The first step in the Gaia analysis process is to find the roles in the system, and the second is to model interactions between the roles found. Roles consist of four attributes: responsibilities, permissions, activities and protocols. Responsibilities are of two types: liveness and safety properties. Role of liveness properties is to add something positive to the system, and role of safety properties is to prevent something detrimental happening to the system. Permissions represents what the role is allowed to do, in particular, which information it is allowed to access. Activities are tasks that a role performs without interacting with other roles. Protocols are the specific patterns of interaction, e.g. a seller role can support different auction protocols, e.g. English auction. Gaia has formal operators and templates for representing roles and their associated attributes, it also has schemas that can be used for the representation of interactions [145].

#### 3.3.2.2 Prometheus

The Prometheus methodology includes three design phases [117], where artifacts are produced which are used in both production of skeleton code for implementation, and for debugging and testing. The system specification phase focuses on identifying the basic functionalities of the system, along with inputs (percepts), outputs (actions) and any important shared data sources. The architectural design phase uses the outputs from the previous phase to determine which agents the system will contain and how they will interact. The detailed design phase looks at the internals of each agent and how it will accomplish its tasks within the overall system.

The Prometheus and INGENIAS methodologies have been customised to take

### 3.3 Agent Design and Development Methodologies and Languages

---

advantage of both approaches in developing agent-based applications [62].

#### 3.3.2.3 Tropos

The Tropos is a software development methodology founded on the key concepts of agent-oriented software development [64]. Specifically, Tropos emphasizes concepts for modelling and analysis during the early requirements phase. Tropos is an agent-oriented software development methodology founded on two novel features. First, the methodology is defined in terms of the concepts of agent, goal, and related mentalistic notions. These notions are used to support all software development phases, from early requirements analysis to implementation. Second, a crucial role is given to early requirements analysis that precedes prescriptive requirements specification. Tropos supports earlier phases of software development compared to other agent and object oriented methodologies.

The Tropos methodology has been extended to enhance its ability to support high variability design through the explicit modelling of alternatives [120]. Moreover, it adopts an extended notion of agent capability and proposes a refined Tropos design process.

#### 3.3.2.4 Multi-agent System Engineering(MaSE)

MaSE is architecture-independent and models a system in terms of goals, roles, agents, tasks, and conversations [46]. Two main stages are included in MaSE: analysis and design. The analysis stage consists of three steps: capturing goals, creating use case and refining roles. The second part of the methodology deals with the design of the system and consists of four steps. The first step [40] is creating agent classes, mapping roles to agent classes in an agent-class diagram. The second step is constructing conversations and defining coordination protocols. The third step

### **3.3 Agent Design and Development Methodologies and Languages**

---

is assembling agent classes and defining the agent architecture and its components. The final step is system designing, creating actual agent instances based on the agent classes which are the output of the previous step.

#### **3.3.2.5 Role Oriented Analysis and Design for Multi-Agent Programming(ROADMAP)**

The ROADMAP methodology extends Gaia and focuses on developing complex open systems giving special emphasis to the societal aspects of the multi-agent system [81]. ROADMAP has features which include support for requirements gathering, explicit models to describe the domain knowledge and the execution environment, levels of abstraction during the analysis phase, the facilitation of iterative decomposition of the system, explicit models and representations of social aspects and individual agent characteristics, from the analysis phase to the final implementation and runtime reflection, and modeling mechanisms to reason and change the social aspects and individual agent characteristics at runtime. ROADMAP consists of two phases: the specification and analysis phase, and the design phase.

### **3.3.3 Object-oriented Based Methodologies**

These kind of methodologies include AAI and AUML. AAI is the Australian Artificial Intelligent Institute and AUML is Agent Unified Modeling Language which is expansion of Unified Modeling Language (UML).

#### **3.3.3.1 Australian Artificial Intelligent Institute (AAI)**

Australian Artificial Intelligent Institute (AAI) originated this methodology and was developed by David Kinny 1996. AAI is based on the BDI paradigm and provides an internal and external prospective of multi-agent systems [21]. AAI builds on

### **3.3 Agent Design and Development Methodologies and Languages**

---

object-oriented models. AAIL provides a specialized set of models: the agent and the interaction models capture the notion of roles, responsibilities, services and control relationships between agents at the external level and the belief, goal and plan models to design BDI agents at the internal level.

#### **3.3.3.2 Agent Unified Modeling Language (AUML)**

The Unified Modeling Language (UML) unifies and formalizes the methods of many object-oriented approaches, including Booch, Rumbaugh (OMT), Jacobson, and Odell [18]. UML is standard language for constructing, visualizing, specifying and documenting the artefacts of a software system, and for business modelling and other non-software systems. Odell, along with colleagues, explored Agent UML (AUML) an extension of UML, to model agent-based applications [20][115]. UML provides an insufficient basis for modeling agents and agent-based systems, and this is due to two reasons: Firstly, compared to objects, agents are active because they can take the initiative and have control over whether and how they process external requests. Secondly, agents do not only act in isolation but in cooperation or coordination with other agents. Multiagent systems are social communities of interdependent members that act individually.

#### **3.3.4 Other Methodologies**

The last class of methodologies includes approaches such as DESIRE and MAS-CommonKADS.

##### **3.3.4.1 Design and Specification of Interacting Reasoning Components(DESIRE)**

The compositional multi-agent design method DESIRE supports the design of component-based autonomous interactive agents [24]. Both the intra-agent functionality and the

### 3.3 Agent Design and Development Methodologies and Languages

---

inter-agent functionality are explicitly modelled. DESIRE supports the conceptual design and specification of both dynamic and static aspects of agent behaviour. DESIRE views the individual agents and the overall system as compositional structures hence all functionality is designed in terms of interacting, compositionally structured components.

#### 3.3.4.2 MAS-CommonKADS

MAS-CommonKADS [77] [46] extends the models defined in CommonKADS, adding techniques from object-oriented methodologies and from protocol engineering to describe the agent protocols [29]. The methodology consists of three main phases. The first phase, called conceptualization, deals with extracting the basic system requirements from the user. In the analysis phase a number of models are developed. These models include agent, task, expertise, and coordination models. In the third phase, called design, the architecture of the system and the individual agents are defined creating the design model.

#### 3.3.5 Programming Languages

There are many languages supporting agent-oriented programming. These programming languages include agent-oriented programming (AOP), AGENT0 programming language, concurrent METATEM, AgentSpeak(L) and APRIL. The AOP is based on a mentalistic view of agents and was first attempted by Shoham, 1993 to create a pure agent-oriented language. It promotes a societal view of computation and allows the direct programming of agents in terms of their mental state. The AGENT0 programming language was developed by Shoham as an implementation of the AOP paradigm. AGENT0 allows the specification of an agent in terms of a set of beliefs, capabilities, commitments, and a set of commitment rules. Concurrent

### 3.3 Agent Design and Development Methodologies and Languages

---

METATEM [51] is a multi-agent system programming language based on linear temporal logic. It consists of currently executing agents whose behaviour is implemented using executable temporal logic and communicate via asynchronous broadcast message passing. AgentSpeak is logic-based approach programming language that allows the specification of BDI agents. It is based on a restricted first order language with events and actions. APRIL is a symbolic programming language that is designed for writing mobile, distributed and agent-based systems. It is compiled into byte code which is then interpreted by the APRIL runtime engine. Besides that, there are some tools and environments which support agent developers which are mostly based on the Java programming language [46]. These include ZEUS toolkit, Java Agent Development (JADE), JACK Intelligent Agents, Java Agent Template (JATLite), Open Agent Architecture (OAA), Foundation for Intelligent Physical Agents-Open Source (FIPA-OS), IBM's Agent Building and Learning Environment (ABLE), and AgentBuilder.

#### 3.3.6 AOP Versus OOP

Object-oriented programming (OOP) is slightly different from agent-oriented programming (AOP). AOP is a specialised form of OOP which enables objects to become agents that have a defined mental state (beliefs, commitments, and capabilities) and send messages to one another using speech act theory. Some modelling techniques and ideas from object-oriented software engineering are adapted because object-oriented modelling techniques are not directly applicable to agent systems, and agents are more complex than objects [74] [116]. Speech act theory categorizes speech, distinguishing between informing, requesting, offering, accepting, rejecting, competing, and so on. Each type of communicative act involves different presuppositions and has effects. The table 3.2 summarizes the relation between AOP and OOP [133].

## 3.4 Multi-agent Systems (MAS)

There is no fixed definition of a multi-agent system, only an agreement on the most common features like multiple agents acting in one environment, each agent having specific goals, communications are between agents themselves and between agents and the environment and actions affect the common environment of all agents in order to solve problems which are difficult or impossible for an individual agent to solve. On the other hand there are some definitions of a multi-agent system [78] which are rather descriptive like: “An agent is a self contained problem solving entity (implemented in hardware, software or a mixture of the two) which exhibits the following properties autonomy, social ability, responsiveness, and proactiveness”.

### 3.4.1 Characteristics of Multi-agent Environments

Multi-agent environments have different characteristics which for example include having infrastructure specifying communication and interaction protocols, open and have no centralized design, having autonomous and distributed agents, flexible and extensible.

### 3.4.2 Applications of Multi-agent Systems (MAS)

There exists many potential industrial and commercial applications for multi-agent systems. Here are some of the applications of multi-agent systems [148]:

- Electronic commerce and electronic markets, where “buyer” and “seller” agents purchase and sell goods on behalf of their users.
- Information handling in information environments like the Internet, where multiple agents are responsible, e.g. for information filtering, gathering, and retrieving.



### 3.5 Agents Role in Grid Computing

---

- Optimization of industrial manufacturing and production processes.
- Analysis of business processes within or between enterprises, where agents represent the people or departments in different stages and at different levels.
- Electronic entertainment and interactive, virtual reality-based computer games, where agents represent characters who play against each other or against humans.
- Social simulation is using agents as an experimental tool. The agents can be used to simulate the behaviour of human societies. Individual agents can be used to represent individual people or organisations.

Other agent applications include:

- User Interface Agents: Microsoft Office Assistant.
- Business process Agents: Data-driven work-flow management.
- Information Management Agents:
  - Email filtering agents.
  - Web browsing assistant.
  - Notification agents.
  - Resource discovery agents.

### 3.5 Agents Role in Grid Computing

Agent technology is increasingly being used in many recent web-based enterprise applications especially for applications dealing with and managing data resources, e.g. data grid. Agent technology is used in computational grid and data grid applications

### 3.5 Agents Role in Grid Computing

---

to enrich them with efficiency and effectiveness. There are many examples of applications, systems, and projects which use agent technology to facilitate the functions of those applications. These examples include:

#### 3.5.1 Grid Resource Discovery

Resource discovery can allow the system to be aware of which resources have been added and which are available. This is done by organization of all information about resources and the status about systems. Available resources in grids usually come from different administrative systems. Using the resource-centric P2P model for grid resource will help to discover resources in the grid environment. To deal with the heterogeneity of allocation semantics within the grid environment, a heterogeneous society of software agents is used. The resource agent layer is a society of agents, these agents respond to receive requests from grid applications and get the corresponding resource from administrative systems (AS), namely deploy grid software on AS [153] [17].

#### 3.5.2 Data Acquisition and Retrieval

Data acquisition and retrieval is used widely nowadays. An example of data acquisition and retrieval is ESESGrid (Engineering Structure Experiment and Simulation Grid) [95]. Data acquisition systems in ESESGrid is based on self-adapting MAS (Multiple Agent System ) and designs the registration and access of agents based on WS-Management. The architecture of ESESGrid consists of four layers, *Local Site*, *Agent Layer*, *Grid Core Service Layer*, and *Client Side*. All agents in the *Agent Layer* are responsible for the real-time experimental data management and access. Those agents collaborate each other, serve each other, resolve conflict between different agents distinct behaviors through competition or negotiation, then complete a

### 3.6 Agent Platforms and Simulators

---

common task. *Agent Layer* makes data acquisition more easy and flexible. This layer consists of two main components, the *Agent Service Adapter* (ASA) for configuring and initializing agents and the *Agent Network Manager* (ANM) that functions as a client to the ASA and helps deploy specific topologies.

#### 3.5.3 Provision Internal Processing in Grid Environment

Agents play a significant role in grid internal processing. An example of agents that provide assistance in grid internal processing is ChinaGrid Support Platform (CGSP). Agent layer has been added to ChinaGrid Support Platform (CGSP) system services [79]. Agents within this layer can be implemented in the Web Services Resource Framework (WSRF) specification as a web service resource in order to extend the autonomy and interoperability for the CGSP system. Two kinds of agents were defined which are Functional Agents and Interoperable Agents. The first kind is Functional Agents which are for the internal processing for grid platform. They have different roles to help CGSP systems managing the resources and running jobs. Functional Agents refer to Provisioning Agent, Execution Agent (Job Agent) and Transferring Agent. The second type is Interoperable Agents which are motivated by the interoperation with other existing heterogeneous grid platforms.

### 3.6 Agent Platforms and Simulators

Agent platforms are used to provide more realistic modeling of agents in the real world. The platform becomes necessary for agents to communicate with each other using appropriate protocols, notify agent's presence to a platform, present common standards for agents to work together, and to produce interoperable multi-agent systems.

There are many agent platforms that have been developed and used to build

### 3.6 Agent Platforms and Simulators

---

multi-agent systems. Those platforms are considered as agent building tools. The platforms are different from each other according to their features, abilities, and common standards.

Table 3.3 summaries main agent platforms and their characteristics:

#### 3.6.1 IBM Aglets Workbench

IBM Japan is developing a framework for Java Agent Applets. Some of their work has been submitted to the Object Management Group (OMG) for consideration in regard to OMGs request for a Mobile Agent Facility (MAF) [88]. Aglets Workbench is a visual environment for building network-based applications that use mobile agents to search for, access and manage corporate data and other information. Aglets Workbench is a very versatile tool for creating secure mobile agent-based applications, however it does not deal with the important issue of implementing coordination, cooperation and coherence in agent-based applications.

#### 3.6.2 Concordia

The Mitsubishi Electric Information Technology Center of America has developed a Java-based framework for development and management of network efficient mobile agent applications for accessing information anytime, anywhere, and on any device [103]. Concordia offers a flexible scheme for dynamic invocation of arbitrary method entry points within a common agent application. It provides support for agent persistence and recovery and guarantees the transmission of agents across a network. Concordia has also been designed to provide for fairly complete security coverage from the outset. Though Concordia provides a useful set of services for implementing agent mobility, security, persistence and transmission , it does not provide any methodology to specify how agents in a multiagent system coordinate, cooperate and

## 3.6 Agent Platforms and Simulators

---

negotiate to bring about a coherent solution.

### 3.6.3 Odyssey

Odyssey<sup>1</sup> is General Magic's initial implementation of mobile agents in Java. It borrows from many of General Magic's concepts in the Telescript tool set. Odyssey is an agent system implemented as a set of Java class libraries that provide support for developing distributed, mobile applications. Odyssey technology implements the concepts of places and agents. It models a network of computers, however large, as a collection of places. A place offers a service to the mobile agents that enter it. A communicating application is modeled as a collection of agents. Each agent occupies a particular place. However, an agent can move from one place to another, thus occupying different places at different times. Agents are independent in that their procedures are performed concurrently. Odyssey provides Java classes for mobile agents and stationary places. Odyssey agents are Java threads. They rely on the same security services as all other Java applications. The developer must adhere to a very rigid structure while implementing mobile applications. Unlike IBM Aglets and Concordia, it does not provide an extensive security mechanism. Also, just like Aglets and Concordia, it is communication centric and does not provide any classes for defining the social behavior of agents. Odyssey does not support broadcast communication and speech-act messaging.

### 3.6.4 Voyager

Voyager<sup>2</sup> is a Java-based agent-enhanced Object Request Broker (ORB) developed by ObjectSpace Inc. It allows Java programmers to quickly and easily create sophis-

---

<sup>1</sup>**Odyssey**, Odyssey Frequently Asked Questions, <http://www.genmagic.com/agents/odyssey-faq.html>, General Magic Inc., 1997.

<sup>2</sup>**Voyager**, Voyager Technical Review, [http://www.objectspace.com/voyager/voyager\\_white\\_papers.html](http://www.objectspace.com/voyager/voyager_white_papers.html), ObjectSpace Inc., 1997.

### 3.6 Agent Platforms and Simulators

---

licated network applications using both traditional and agent-enhanced distributed programming techniques. It provides for creation of both autonomous mobile agents and objects. Voyager agents roam a network and continue to execute as they move. Voyager can remotely construct and communicate with any Java class, even third party libraries, without source. It allows seamless support for object mobility. Once created, any serializable object can be moved to a new location, even while the object is receiving messages. Messages sent to the old location are automatically forwarded to the new location. Voyager is a very efficient tool for constructing agent-based distributed applications, but it suffers from the same drawbacks as Aglets, Concoridia, and Odyssey. It does not provide any classes for defining the social behavior of agents. Voyager does not support broadcast communication and speech-act messaging. It does not pay any specific attention to providing security.

#### 3.6.5 JATLite

JATLite<sup>3</sup> is being developed by the Computer Science Department at Stanford University. It provides a set of Java packages that facilitates agent framework development using the Java language. JATLite supports Speech Acts and provides basic communication tools and templates for developing agents that exchange KQML messages through TCP/IP. It defines a special construct called an *Agent Name Server* (ANS) which stores all the names and addresses of existing agents. When an agent is created and connected to the network, it first registers with the ANS. In registering, the agent passes the ANS its name, port number and the domain of its local host. If an agent knowingly terminates, it first sends a remove address message to the ANS, which echoes the message to all the other agents. JATLite also provides special *Agent Router* functionality which allows Java applets to exchange messages with any reg-

---

<sup>3</sup>**JATLite**, JATLite overview, [http://java.stanford.edu/java\\_agent/htmlJATLiteOverview.html/](http://java.stanford.edu/java_agent/htmlJATLiteOverview.html/), Stanford University, 1997.

## 3.6 Agent Platforms and Simulators

---

istered agent on the Internet. Though JATLite does provide essential functionality required for building a multiagent application, it does not define a methodology for specifying the social behavior of agents. Moreover, the concepts of the ANS and the Agent Router are inherently centralized in nature. Each time an agent joins the system it has to register with ANS and when it leaves the system, the ANS has to be informed. All communication must go through the *Agent Router*. Thus, any application developed using JATLite cannot be truly scalable.

Following is a list of major, publicly available implementations of agent platforms which conform to the FIPA Specifications:

### 3.6.6 Agent Development Kit

The author of Agent Development Kit is a private company called Tryllian BV. Tryllian introduces the latest release of the Agent Development Kit (ADK), a mobile component-based development platform that allows you to build reliable and scalable industrial strength applications [53]. The ADK features dynamic tasking, JXTA-based P2P architecture with XML message-based communication that supports FIPA and SOAP, Java Naming and Directory Interface (JNDI) directory services, using a reliable, lightweight runtime environment based on Java. These allow Java developers to easily build, deploy and manage secure, large-scale distributed solutions that operate regardless of location, environment or protocol, enabling an adaptive, dynamic response to changes. The ADK runs on any environment supporting Java 2 Standard Edition version 1.3.1. A subset of its functionality is available for J2ME MIDP (Mobile Information Device Profile). There is a commercial license and free research license available for selected projects.

## 3.6 Agent Platforms and Simulators

---

### 3.6.7 April Agent Platform

The authors of the April Agent platform are Jonathan Dale and Johnny Knottenbelt. The April Agent Platform (AAP) is a FIPA-compliant agent platform that is designed to be a lightweight and powerful solution for developing agent-based systems [53]. It is written using the April programming language and the InterAgent Communication System (IMC), and provides many features to accelerate the development and deployment of agents and agent platforms. The AAP requires the April programming language and the ICM to be installed, and runs either on Linux, Unix or Windows. It is GNU General Public License (GPL).

### 3.6.8 Comtec Agent Platform

The author of the Comtec Agent platform are Information-Technology Promotion Agency, Japan and Communication Technologies. The Comtec Agent platform is an open-source platform, compatible with FIPA agent communication, which manages agent message transport and some other applications [53]. Unique to the Comtec Platform is the implementation of FIPA Ontology Service and Agent/Software Integration, which require Semantic Language (SL2) as the content language. It requires JDK 1.2 or higher and it is GNU General Public License (GPL).

### 3.6.9 FIPA-OS

The author of FIPA-OS is software development company with expertise in intelligent agents called Emorphia with association of other contributors. FIPA-OS was the first Open Source implementation of the FIPA standard and has now recorded thousands of downloads [53]. Dedicated developers from around the world have contributed to numerous bug fixes and upgrades, leading to over 10 formal new releases. FIPA-OS now supports most of the FIPA experimental specifications currently un-



## 3.6 Agent Platforms and Simulators

---

der development. With the new in depth developer guides, it is an ideal starting point for any agent developer wishing to benefit from FIPA technology. FIPA-OS 2 is a component-based toolkit implemented in 100% pure Java. One of the most significant contributions received is a small, footprint version of FIPA-OS, aimed at Personal Digital Assistants (PDAs) and smart mobile phones, which has been developed by the University of Helsinki as part of the IST project, Crumpet. It requires Java virtual machine and the license is Eclipse Public License (EPL).

### 3.6.10 Grasshopper

Grasshopper is an open 100% Java-based mobile intelligent agent platform, which is compliant to both available international agent standards, namely the OMG MASIF (Mobile Agent System Interoperability Facilities) and FIPA specifications [53]. Grasshopper includes two optional open source extensions providing the OMG MASIF and FIPA standard interfaces for agent/platform interoperability. It requires java virtual machine.

### 3.6.11 JACK Intelligent Agents

The authors of JACK Intelligent Agents are Autonomous Decision-making Software and Agent Oriented Software AOS Ltd (AOS Group's UK company)<sup>4</sup>. It is the world's leading autonomous systems development platform. It has a proven track record in the development of applications that interact with a complex and ever-changing environment. JACK consists [26] of architecture-independent facilities, plus a set of plug-in components that address the requirements of specific agent architectures. The plug-ins supplied with version 1.2, released at the end of October 1998, include support for the BDI model.

---

<sup>4</sup>Autonomous Decision-making Software AOS Ltd, <http://www.agent-software.com/products/index.html>.

## 3.6 Agent Platforms and Simulators

---

### 3.6.12 JAS (Java Agent Services API)

The authors of JAS are Fujitsu, Sun, IBM, HP, Spawar, InterX, Institute of Human and Machine Cognition, Comtec and Verizon. The Java Agent Services (JAS) project defines an industry standard specification and API for the deployment of agent platform-service infrastructures [53]. It is an implementation of the FIPA Abstract Architecture within the Java Community Process initiative and is intended to form the basis for creating commercial grade applications based on FIPA specifications. Specifically, the project consists of a Java API (in the `javax.agent` namespace) for deploying open platform architectures that support the plug-in of third-party platform service technology. The API provides interfaces for message creation, message encoding, message transport, directory and naming. This design is intended to ensure that a JAS based system deployment remains transparent to shifts in underlying technology without causing interruption to service delivery and therefore the business process. The project also delivers a Reference Implementation of the API, including sample services for Remote Method Invocation (RMI), Lightweight Directory Access Protocol (LDAP), and HTTP. The forthcoming Technology Compatibility Kit will ensure compliance of all JAS based implementations. It requires java virtual machine (1.1 minimum). It is Open Specification License v0.4 and Common Public License v0.5.

### 3.6.13 ZEUS

ZEUS is an Open Source agent system entirely implemented in Java, developed by BT Labs and can be considered a toolkit for constructing collaborative multi-agent applications. ZEUS provides support for generic agent functionality and has sophisticated support for the planning and scheduling of an agent's actions [53]. Moreover, ZEUS provides facilities for supporting agent communications using FIPA ACL

## 3.6 Agent Platforms and Simulators

---

(Agent Communication Language) as the message transport and TCP/IP sockets as the delivery mechanism. ZEUS also provides facilities for building agents in a visual environment and support for redirecting agent behavior. The ZEUS approach to planning and scheduling involves representing goals and actions using descriptions that include the resources they require and the pre-conditions that need to be met in order to function. This allows goals to be represented using a chain of actions that have to be fulfilled before the goal can be met. This action chain is built up using a process of backwards chaining. As ZEUS uses the latest Swing GUI components it will run on any platform that has had a JDK1.2 (aka JDK2) virtual machine installed. Each host machine should also be capable of TCP/IP communication, but there is no need for any middleware services to be installed. So far ZEUS has been successfully tested on Windows 95/98/NT4 and Solaris platforms.

### 3.6.14 JADE Platform

JADE platform has many features which have encouraged us to choose it as the platform of our prototype. JADE has the ability to build agents-based distributed systems which are suitable for grid systems, agents have the ability to control other system components, heterogeneous entities communication, security, and interoperability with other agents. JADE is the middleware developed by TILAB<sup>5</sup> for the development of distributed multi-agent applications. The intelligence, the initiative, the information, the resources and the control can be fully distributed on mobile terminals as well as on computers in the fixed network [44].

JADE is implemented in version 1.2 of JAVA (Jade 3.7 latest version July 2009) and has no further dependency on third-party software. It requires Java virtual

---

<sup>5</sup>Telecom Italia Lab is the R&D branch of the Telecom Italia Group and is responsible for promoting technological innovation by scouting new technologies, carrying out and assessing feasibility studies, and developing prototypes and emulators of new services and products. Telecom Italia has conceived and developed JADE, and originated the Open Source Community in February 2000.

### 3.6 Agent Platforms and Simulators

---

machine (1.2 minimum) and it is Lesser General Public License (LGPL). JADE is an ideal platform on which to implement our model in order to present concepts and objectives of our research. In [32], authors have proven some features of the JADE platform. These features include efficiency, effectiveness, and scalability. The features are limited by standard limitations of Java programming language and other factors which include processor speed, amount of available memory and speed of network connection. Experiments with thousands of agents and thousands of ACL messages have been implemented effectively. Hence, these features of the JADE platform are needed in our model in order to present concepts and objectives of our research.

The JADE platform is composed of agent containers that can be distributed over a network [19]. Agents live in containers which provide the JADE run-time and all the services needed for hosting and executing agents. There is a special container, called the “Main Container”, which represents the bootstrap point of a platform. The main container is the first to be launched and all other containers must join to it by registering with it. As a bootstrap point, the main container has the following special responsibilities:

- Managing the container table (CT), which is the registry of the object references and transporting addresses of all container nodes composing the platform.
- Managing the global agent descriptor table (GADT), which is the registry of all agents present in the platform, including their current status and location.
- Hosting the Agent Management System (AMS) and the Directory Facilitator (DF), the two special agents that provide the agent management and white page service, and the default yellow page service of the platform, respectively.

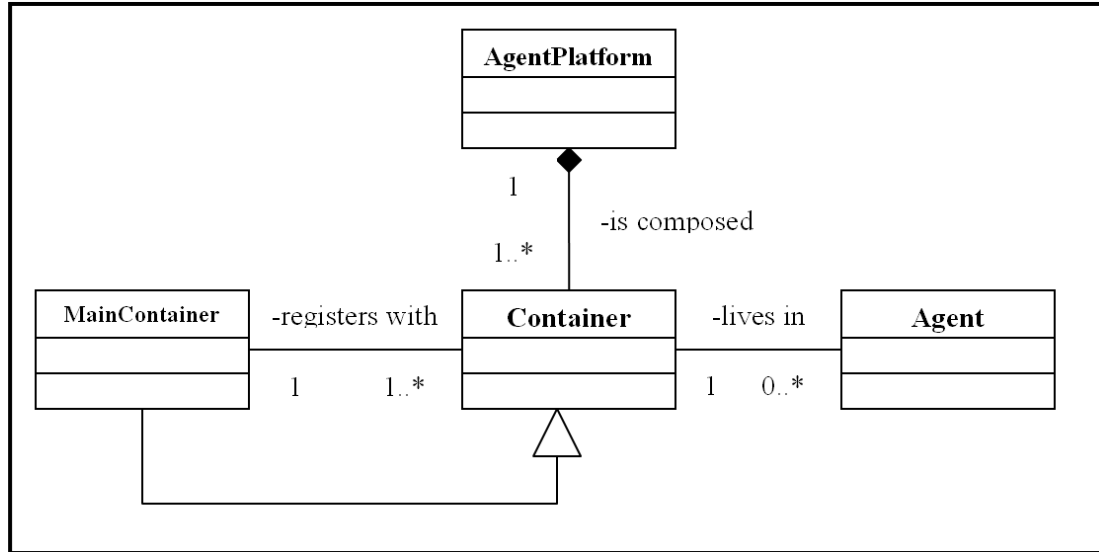
The Agent Management System (AMS) provides the naming service (i.e. ensures that each agent in the platform has a unique name) and represents the authority in

### 3.6 Agent Platforms and Simulators

the platform (for instance it is possible to create/kill agents on remote containers by requesting that to the AMS). According to the FIPA architecture, this is the agent that is responsible for managing the platform and providing the white-page service.

The DF provides a Yellow Pages service by means of which an agent can find other agents providing the services it requires in order to achieve its goals. According to the FIPA architecture, this is the agent that provides the yellow-page service.

Figure 3.2 shows the relationship between the main architectural elements of JADE using a UML diagram [19].



**Figure 3.2:** UML JADE Architecture

#### 3.6.15 Lightweight Extensible Agent Platform (LEAP)

LEAP (Lightweight Extensible Agent Platform (IST-1999-10211)) is a development and run-time environment for Intelligent Agents, and is the precursor of the second generation of FIPA compliant platforms [53]. It represents a major technical challenge - it aims to become the first integrated agent development environment capable of generating agent applications in the ZEUS environment and executing

## 3.6 Agent Platforms and Simulators

---

them on run-time environments derived from JADE, implemented over a large family of devices (computers, PDA and mobile phones) and communication mechanisms (TCP/IP, WAP). In this way LEAP benefits from the advanced design-time features of ZEUS and the lightweight and extensible properties of JADE. It requires java virtual machine.

### 3.6.16 Agent Platforms Comparison

Agent platform is a technological architecture which provides the environment in which agents can exist and operate in order to achieve their goals. The agent platform may additionally support the development of agents and agent-based applications.

Table 3.4 is a comparison study of some agent platforms. The criteria has been determined in the study based on characteristics of optimal e-learning system (RDADeLE system) to come up with a dependable comparison study. They are 4 criteria which include high performance, policy enabled, FIPA-compliant and heterogeneity.

The comparison study includes 20 different agent platforms and simulators. These agent platforms and simulators are different in the programming language in which they are written in, the purpose and needs of users and characteristics. From the table 3.4 although specifications of the Practionist and the JADE are similar there are two differences. The first one is that the Practionist works on top of the JADE and Prolog which means that the Practionist depends on the JADE. The other difference is that the Practionist produces BDI agents only. Moreover, JADE is well supported (documentation, mailing list, platform updates), free and popular platform. It is important that an agent platform foresees the use of mobile devices to have a light-weighted release. The JADE platform has a light-weighted release which is called Lightweight Extensible Agent Platform (LEAP).

## 3.7 Review of Agent Technology towards E-learning Requirements

In this chapter, we have presented agent technology. The first major requirement is management. Part of the management is monitoring all components of the e-learning architecture. Indeed, monitoring components of the e-learning architecture is indispensable because to keep the architecture must adapt itself according to changes in the e-learning environment. Using agent technology in the prototype of e-learning promises better results in monitoring its components. It has been show in [92] that the Monalisa distributed system is monitored, controlled and optimized using agent technology. Therefore agent technology is the choice for monitoring components in the architecture.

The third major requirement of the architecture is flexibility. Flexibility refers to the composing and searching for LOs's information and services in the context architecture. Two techniques, XML-based Registries Technique (XRT) and Registered-based Services Technique (RST), have been developed and implemented to compose and search for LOs'information and services in chapter 7. The difference between two techniques is that the data source information in the second technique is published as service agents. The result of conducting two techniques shows that using the RST technique the RDADeLE environment can be built with more regional grids with less memory consumption. Meanwhile, It shows that more grid services could be registered in the RDADeLE system with less mean search time using the RST technique. The search performance increases using the RST technique compared with the XRT technique.

The fourth major e-learning requirement is fault tolerance. Fault tolerance is a crucial issue in distributed and dynamic systems. The e-learning architecture, as an

### 3.8 Summary

---

e-learning system, needs to bind all administrative regions of the Kingdom of Saudi Arabia with each other and with their components in order to maintain reliability to become more robust and reliable. Choosing an appropriate agent platform to assist the architecture to become more reliable and robust is strongly recommended. [32] and [82] shows that JADE agent platform is efficient, scalable and reliable. Furthermore, using the JADE agent platform to generate the required agents in the e-learning system is recommended in order to monitor the environment agents and then help the e-learning architecture to become more flexible and reliable. There are many R&D (Research and Development) projects have used and using JADE. For example, these are some projects use the JADE: AgentCities, TeSCHeT, PRIMO, IMAGE, MicroGrids, E-Commerce Agent Platform (E-CAP). The conclusion is that agent technology can support the flexibility and fault tolerance requirements of the e-learning architecture, with the JADE as the agent platform.

### 3.8 Summary

This chapter surveyed the background to and related work on agent-base computing. It described agent technology, including classification of agents, agent architectures, and agent interaction protocols. This is followed by a survey of agent design and development. Multi-agent systems are then described. Afterwards, a survey of agents role in grid computing is presented. Finally, agent platforms and simulators are presented with a comparison study between them.



### 3.8 Summary

---

**Table 3.2:** OOP versus AOP

Basic unit	OOP object	AOP agent
Parameters defining state of basic unit	unconstrained	beliefs, commitments, capabilities, choices...
Process of computation	message passing and response methods	message passing and response methods
Types of message	unconstrained	inform, request, offer, promise, decline...
Constraints on methods	none	honesty, consistency...

**Table 3.3:** Main Characteristics of Major Agent Platforms

Platform	Company	URL	Main characteristics
ADK	Tryllian	<a href="http://www.tryllian.com">www.tryllian.com</a>	BPM focused
Enago	IKV++	<a href="http://www.ikv.de">www.ikv.de</a>	Mobile; academic
FIPA-OS	emorphia	<a href="http://www.emorphia.com">www.emorphia.com</a>	Meeting scheduler
Jack	AOS	<a href="http://www.agent-software.com">www.agent-software.com</a>	Generic; academic
Jackdaw	Calico Jack	<a href="http://www.calicojack.co.uk">www.calicojack.co.uk</a>	Generic; mobile focus
Jade	Telecom Italia	<a href="http://jade.tilab.com">jade.tilab.com</a>	Generic; academic
ZEUS	BT	<a href="http://www.bt.com">www.bt.com</a>	Research oriented

### 3.8 Summary

---

**Table 3.4:** Comparisons of Some Agent Platforms

Agent Platforms	High Performance	Policy Enabled	FIPA-compliant	Heterogeneity	Security
JACK	Yes	Yes	Yes	n/a	n/a
ADK	n/a	n/a	Yes	No	Yes
Cougaar	n/a	n/a	No	Yes	Yes
Cybele	Yes	Yes	No	n/a	n/a
Agent Factory	n/a	Yes	Yes	Yes	Yes
3APL	n/a	n/a	Yes	No	No
FIPA-OS	n/a	n/a	Yes	Yes	No
AgentBuilder	Yes	n/a	No	No	n/a
MadKit	n/a	No	No	n/a	Yes
DIET Agents	Yes	n/a	No	n/a	Yes
JIAC	No	n/a	Yes	n/a	Yes
SAGE	Yes	n/a	Yes	n/a	Yes
Lost Wax	Yes	Yes	n/a	Yes	n/a
Semoa	n/a	Yes	Yes	Yes	Yes
A-Globe	Yes	n/a	No	No	n/a
ABLE	Yes	Yes	Yes	n/a	Yes
Praxionist	Yes	Yes	Yes	Yes	Yes
AgentScape	Yes	n/a	n/a	Yes	Yes
ZEUS	Yes	n/a	Yes	Yes	n/a
JADE	Yes	Yes	Yes	Yes	Yes

## Chapter 4

# Architectural Design of Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE)

### Objectives

---

- Review requirement analysis.
  - Present motivation for building the RDADeLE system.
  - Discuss the computational model of the RDADeLE system.
  - Present contents and services management.
  - Discuss fault-tolerance with replicated information service in the RDADeLE system.
- 

## 4.1 Introduction

The RDADeLE system has been constructed for controlling and managing large systems for instance systems of systems and e-learning systems. Although there are an increasing number of e-learning systems nowadays, some of these systems still have hidden technical problems which remain undiscovered. Unfortunately, many specialists who are responsible for dealing with and enhancing these systems are not aware of some practical barriers and problems which prevent gaining full benefits of using them. These practical problems include an increase in information flow, accumulation of work in specific points and a loss of connections between different parties and

## 4.2 Requirement Analysis

---

elements within each system and between systems. It is essential that these problems are solved, and one of the ways to solve them is to embed different technologies such as grid and agent in the RDADeLE system.

The grid community focuses on brawn: infrastructure, tools, and applications for reliable and secure resource sharing within dynamic and geographically distributed virtual organizations. In contrast, the agents community focuses on brain: autonomous problem solvers that can act flexibly in uncertain and dynamic environments. Agent systems require robust infrastructure and grid systems require autonomous, flexible behaviors [57]. Thus systems based on agent and grid technology are capable of controlling and managing their components in order to become reliable, scalable and dynamic.

This new e-learning architecture combines both grid and agent technologies to produce the RDADeLE system. This can be done by defining different classes of agents comprising *the Administrative Agent* which works as a grid information service and other agents used to control components of the e-learning system. The latter are *the Regional Agent*, *the Node Agent*, *the Learner Agent* and *the Service Agent*. This chapter presents requirement analysis, computational model which is guided by a case study, the architecture of RDADeLE and its components, contents and services management and fault-tolerance.

## 4.2 Requirement Analysis

Designing an effective architecture requires thorough analysis. Users and environment needs are the most important factors to be considered. The following are highlights of analysis taken as guidelines for producing the RDADeLE architecture:

## 4.2 Requirement Analysis

---

### 4.2.1 Previous e-Learning Architectures

By reviewing many papers in the field of e-learning, there are numerous architectures and platforms which have been adopted as testbeds and prototypes. One of these is a Learning Management System based on the Life Cycle Management Model (LCMM) of e-learning courseware [71]. This model concentrates on analysis, design, development, delivery and measurement of courseware content activities known collectively as Sharable Content Object Reference Model(SCORM) - within the e-learning environment. This architecture does not include collaboration with other learning management systems on other sites. Another e-learning architecture was proposed based on workflow using fuzzy Petri nets [84]. This architecture takes account of the processes of academic study activities which means that it navigates the learning resources (i.e. Learning Objects (LOs)), adapting to each learners pace by formal description of their learning path and the application of workflow technology in building and implementing workflow of courses in the service environment. Also, this architecture does not consider the LO service as a grid service. Another architecture called an Architecture of Virtual Environment for E-Learning (AVEE) which depicts the virtual environment of e-learning and integrates a media stream into the learning environment (virtual reality); it is based on the MVC architecture (Model, View Controller) [73]. This architecture limits its usage between institutes and sites.

An approach for designing an adaptive multi-agent architecture of e-learning system has been adopted in [69]. The design applies various technologies, integrating them in a flexible and efficient way to support their adaptation in e-learning environments. The approach is based on an intelligent blackboard model for content presentation and learning strategy planning. The intelligent blackboard includes three levels of interfaces through which the structure of e-learning courses can be defined. This approach does not include data grid in its architecture and does not

## 4.2 Requirement Analysis

---

intend to control and manage the e-learning system as a system of systems.

In [70] an agent-based design for a learning environment using the Prometheus methodology and JACK platform has been presented. The architecture is divided into three spaces which are individual space, collaborative space and cooperative space. The system evaluates its impacts at different levels such a pedagogic, social and economic. The system depicts the epistemic and cognitive profile of the learner to adapt according to its capacities. This system does not use data grid in its architecture and does not intend to control and manage the e-learning system as a system of systems.

Modelling of human learners using agents has been investigated in [132]. Agents can facilitate personalised learning. Agents, which replaces the human instructors, control an e-learning environment and exploits a self organising map (SOM) in order to achieve the learning goal. The goal of the study was to support the argument that e-learning environments are feasible and can significantly assist dissemination of knowledge within modern educational settings.

In [25] the author proposed an agent-oriented extensible framework based on Extensible Markup Language (XML) for building a hypermedia e-learning system available on the World Wide Web. The implementation focused on deploying mobile agents that can exchange information in a flexible way via XML-based documents such as RDF assertions and SOAP messages.

The Multi-Agent System for E-Learning and Skill Management (MASEL) was described in [61]. The MASEL performs tasks which include for example (1) supporting Chief Learning Officers in defining roles, associated competencies and required knowledge level; (2) managing the skill map of the organization; (3) evaluating human resources' competence gaps; (4) supporting employees in filling the competence gaps related to their roles; (5) creating personalised learning paths according to feedback

## 4.2 Requirement Analysis

---

that users provide to optimise the acquisition of required competencies. A prototype was developed using JADE (Java Agent DEvelopment Framework).

In [6] the author proposed architecture for an e-learning system based on mobile agent technology. The learner's actions were monitored by a mobile agent in the e-learning system. Two functions were performed: (1) the mobile agent identified optimal learning conditions and (2) noted the areas of weak knowledge. The proposed architecture focused on the process of composing personalised content for an individual user and developing courses.

### 4.2.2 Context and Motivation

With the emergence of the internet as the backbone of global communication and information exchange, greater attention has been paid to e-learning. Many academic institutions and training centres worldwide are embracing web-delivered instruction. In recent years, new public and private universities have been established to offer degree programs delivered exclusively online. In the Kingdom of Saudi Arabia (KSA) [12] there are many reasons to adopt and embrace the RDADeLE architecture, the following are some of those reasons:

1. The KSA covers an area of 2,149,690 sq.km, and is divided into 13 administrative regions (See table 4.1 and map figure 4.1). This large area of different administrative regions requires systems to control and manage them. The administrative regions of the KSA [1] are based on the regional government system and are regarded as the basis for collection, distribution, and publication of geographical and statistical data. The RDADeLE system needs to be divided into many segments (regions) according to the administrative regional divisions. The division is not provided for division sake only, it is provided to minimise problems of the whole project by determining and solving problems associated

## 4.2 Requirement Analysis

---

with each segment. Besides that, division is provided to build autonomous segments. Eventually, the system will combine these segments to build the whole system. This approach has been embraced to build the required RDADeLE system. The segments in the RDADeLE system are supposed to be regional grids which correspond to the administrative regions of the KSA.

**Table 4.1:** Administrative Regions of the Kingdom of Saudi Arabia

No.	Administrative Region	Headquarter
1	Riyadh Region	Riyadh City
2	Makkah Region	Makkah City
3	Madinah Region	Madinah City
4	Qasim Region	Buraydah City
5	Eastern Region	Dammam City
6	Asir Region	Abha City
7	Tabouk Region	Tabouk City
8	Hail Region	Hail City
9	Northern Border Region	Arar City
10	Jizan Region	Jizan City
11	Najran Region	Najran City
12	Baha Region	Baha City
13	Al-Jouf Region	Sikaka City

- Information: The KSA has 252 government and private universities and colleges that grant bachelor, master and doctorate degrees in different fields (See table 4.2). The increasing number of universities, colleges and institutions will result in massive amounts of information on the web assuming that education will be delivered via the internet as well as traditional methods. This information should be exploited by organising and sharing it between different environments of universities, colleges and institutions using distributed regional grids.
- Demographics: According to a United Nations study in 2006 [108], the world's population is increasing; the KSA certainly follows this trend. Table 4.3 illus-



## 4.2 Requirement Analysis



**Figure 4.1:** Administrative Regions in the Kingdom of Saudi Arabia

**Table 4.2:** Distribution of Universities, Colleges and Institutions Among Administrative Regions of the Kingdom of Saudi Arabia

Administrative Region	Number of Institutions	Number of students
Riyadh Region	57	183807
Makkah Region	40	178428
Madinah Region	16	46540
Qasim Region	22	32289
Eastern Region	30	76798
Asir Region	22	49745
Tabouk Region	9	11354
Hail Region	8	12487
Northern Border Region	5	9244
Jizan Region	10	24375
Najran Region	9	9344
Baha Region	10	14696
Al-Jouf Region	14	17555
Total	252	666662

## 4.2 Requirement Analysis

---

trates these changes. This means that more universities, colleges and institutions will open.

**Table 4.3:** World and the Kingdom of Saudi Arabia Population (thousands)

Year	2007	2015	2025	2050
World	6671226	7295135	8010509	9191287
Saudi Arabia	24735	29265	34797	45030

The education system in the KSA is divided into three main categories: general education, technical and vocational education, and higher education [4]. General education is under the supervision of the Ministry of Education, which was established in 1954. General education consists of six years of elementary school, beginning at the age of six, three years of intermediate and three years of general secondary school. Technical and vocational education is under the supervision of the Technical and Vocational Training Corporation (TVTC) which has more than 40 technical colleges and more than 120 other technical and vocational institutions [142]. Higher education is under the supervision of the Ministry of Higher Education. The number of registered students in 2009 in universities and colleges under the umbrella of higher education and technical and vocational education, was 666662 students, and the number of faculty staff was 27964 members [28]. These figures indicate an increase in the number of students and faculty staff in universities, colleges and institutions in the KSA. Table 4.2 shows the distribution of universities, colleges, institutions and students among the 13 administrative regions of the KSA.

We concentrate on all institutions which are supervised by the Ministry of Higher Education. We can adopt the RDADeLE system to be adapted and deployed to control and manage universities, colleges, faculty staff and students

## 4.2 Requirement Analysis

---

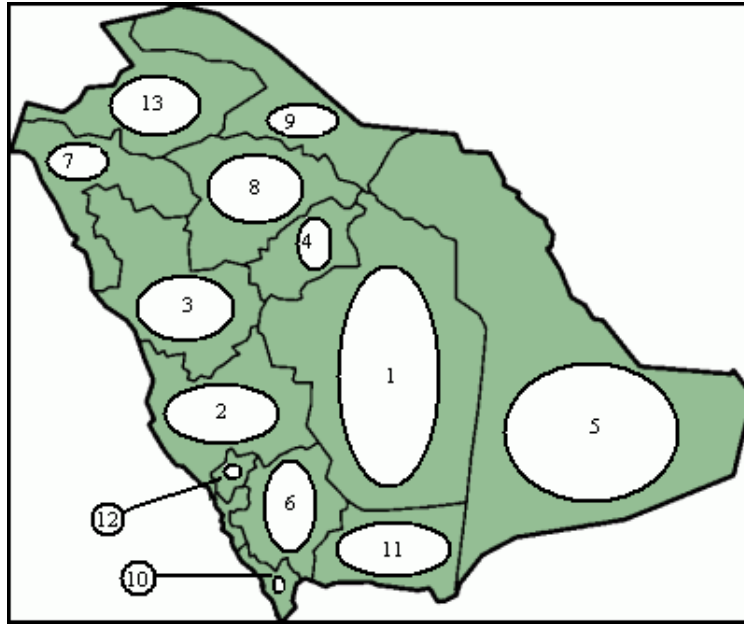
of this category within the administrative regions of the KSA.

4. Economics: According to the International Monetary Fund's World Economic Outlook Report, April 2009, the KSA and 154 more countries were considered as developing countries which have been described as nations with a low level of material well being. Controlling and managing e-learning environments will encourage education institutions to use e-learning technology. Using e-learning technology will reduce the number of students who use traditional classes in academic institutions to pursue their studies. Therefore, budgets dedicated to running institutions and training centres will be reduced. In the economical context, internal migration to big cities will be limited if e-learning is adopted by academic institutions. Using e-learning will actively prevent young people from migrating from rural areas to cities, considering that communication technology is accessible in most inhabited areas. Preventing internal migration of people from rural areas to big cities will increase the economy of these areas which in turn will increase the economy of the whole country.
5. Society and culture: In the KSA, King Faisal faced public resistance against female education. He was not initially able to convince people of his views [68]. In 1963, King Faisal commanded the official force to keep girls' schools open. Although there is not a single verse in the Holy Quran which forbids the education of women, some parents prevented their daughters or sisters from pursuing higher or even secondary education. This was on account of some local tribal traditions which mandated segregation of genders [75] and social conservatism. An increase in using and adopting e-learning architectures in academic institutions will generously help tackle these problems. Hence, girls can pursue higher education using the web without attending actual classes.

## 4.2 Requirement Analysis

---

These statistics related to the education system in the KSA motivates us to build a robust system in order to organise, control and manage e-learning in the KSA. Figure 4.2 shows the deployment of the RDADeLE system in the administrative regions in the KSA. The map shows that there are 13 regional grids corresponding to the number of the administrative regions (See table 4.1).



**Figure 4.2:** RDADeLE System Deployment in the Administrative Regions in the Kingdom of Saudi Arabia

### 4.2.3 Requirements for Dynamic RDADeLE Environment

In order to produce such an effective and dynamic model to tackle issues related to controlling and managing complex and large scale systems, we consider three main criteria which must be adopted in the model.

The first criteria is *the adaptability* which is an important factor in dynamic and complex systems. Adaptability is the ability of the RDADeLE system to adapt itself efficiently and quickly to changing circumstances. An adaptive system is able to adapt its behaviour according to changes in its environment or in parts of the system itself.

## 4.2 Requirement Analysis

---

Since there are so many parties and components that affect and contact with the e-learning environment, RDADeLE needs to adapt to these changes. Agent features will help the systems to be adaptable.

The second criteria is *the flexibility*, which is another factor needed in dynamic systems. Flexibility is the ability of a system to respond to potential internal or external changes affecting its value delivery, in a timely and cost-effective manner. Thus, flexibility for the RDADeLE system is the ease with which the system can respond in a manner to increase such value delivery. Embedded agents in the RDADeLE system will help the system to be flexible.

The third criteria is *the scalability*, a factor that a dynamic system depends on. In the RDADeLE architecture, the data grid has been divided into regional grids. The regional grids with its middlewares (e.g. OGSA and OGSF) are responsible for scalability, reliability and integration of data grids within a grid environment. A large-scale grid system can create added value such as connecting large numbers of resources, allowing them to be utilised and thereby enabling new services [54].

Communication within the RDADeLE system is another important factor. Use of agent technology to resolve this problem is highly recommended; this led us to produce a dynamic architecture.

### 4.2.4 The Architectural Data Standards in the RDADeLE System

Data sharing has become an increasingly important aspect within the data grid environment. Many systems face the critical challenge of sharing information. This issue is amplified in the system of systems where the components, e.g. regional grids, require data exchange to operate and for the overall system of systems to work. Data standards are fundamental to the seamless exchange of data and they help improve

## 4.2 Requirement Analysis

---

the ability to exchange data efficiently and accurately. They also assist data users to understand, interpret, and use data appropriately. Data standards improve the quality and ability of sharing in complex and system of systems environments by increasing data compatibility, improving the consistency and efficiency of data collection and reducing data redundancy. In the RDADeLE system, we adopted the Sharable Content Object Reference Model (SCORM) to design the content distribution of Learning Objects (LOs) in the regional grids and the Dublin Core Metadata Element Set (DCMES) to design the index of the LOs (i.e. LOs' information).

### 4.2.4.1 Standards of Learning Objects (LOs) Content Distributions

The Learning Object (LO) was defined by the Learning Technology Standard Committee (LTSC) of the Institute of Electronic and Electronic Engineers (IEEE) as “any entity, digital or non digital, that may be used for learning, education or training” [37]. It was noted that LOs are extensively used by well known corporations such as Cisco and Microsoft. AT&T Business Learning Services also adopt this technology for internal and customer training [80] [35].

Several standards have been developed for the purpose of learning objects' content distributions. There are some common standards used in the e-learning field. These standards include Sharable Content Object Reference Model (SCORM) which was generated by the Advanced Distributed Learning (ADL) initiative. ADL has made rapid progress via incorporation due to the efforts of the Instructional Management Standards (IMS), The Aviation Industry CBT (Computer-Based Training) Committee (AICC), the Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE), and the IEEE Learning Technology Standards Committee into a single harmonised reference model for learning design and delivery.

### 4.3 The Computational Model of The RDADeLE System

---

#### 4.2.4.2 Metadata Standards

When we talk about LOs, we usually refer to metadata. Metadata is used to make LOs reusable, sharable, storable and manageable in a repository such as an archive. The most popular metadata standards that provide cataloguing, searching and reuse of resources are the Dublin Core Metadata Element Set (DCMES) and the IEEE Learning Object Metadata (LOM). Nowadays, search engines use catalogue concepts to structure web page contents. Having information about the content makes it easier for humans and computers to classify a resource.

On one hand, the IEEE LOM organises its 60 elements into nine categories: *general, life cycle, meta-metadata, technical, educational, rights, relation, annotation* and *classification*. On the other hand, the current version of DCMES comprises 15 well-defined elements for describing the core information properties: *title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage*, and *rights* [101].

### 4.3 The Computational Model of The RDADeLE System

For many years, learning methods have been traditional, considering teachers who are providers and transmitters to learners of educational information, as the centre of the learning process [59]. Transferring traditional learning methods to e-learning and web based technologies plays a significant role in the learning and teaching process. e-Learning is the employment of technology to aid and enhance learning. It can be as simple as high school students watching a video documentary in class or as complex as an entire university course provided online [41].

In order to present a clear view of the RDADeLE system we will present its

### 4.3 The Computational Model of The RDADeLE System

---

computational model which will be guided by the KSA case study. Our computational model comprises entities and mechanisms. Entities include *regional grids*, *nodes*, *learners*, *data resources* and *policy*, while mechanisms include *communication*, *timing* and *failure mechanism*.

#### 4.3.1 The Entities

Entities are the main components of our computational model which include *regional grids*, *nodes*, *data resources* and *policy*. Each of these entities is an object as follows:

1. *Regional grids*: Regional grids are the entity which comprise the entire grid in a distributed fashion. In our case study there are 13 regional grids corresponding to the 13 administrative regions in the KSA. Each regional grid is supposed to be a “container” which keeps track of and connections between the regional grid’s components. From the agent platform point of view the “container” is the place where agents live. Regional grid components include nodes, learners, grid services, data and policy. The Regional grid is responsible for the activities within its area using some grid middlewares. One of these activities which we are concerned about in the case study in our research is search provision. The search process is for LOs’ information and grid services. The search process is performed by *the Regional Agent*, which manages the regional grid, responding to requests from learners within or outside the regional grid. Moreover, regional grid’s agents manage its components to show some kind of autonomy.
2. *Nodes*: Nodes are the entities which manage learners connected with them using *the Node Agent* within a regional grid. Local Area Network (LAN) and Wide Area Network (WAN) are network capabilities which are used for communication between nodes. Capabilities and roles of a node depend on



### 4.3 The Computational Model of The RDADeLE System

---

the type of node in each regional grid. There are two types of node which are computation node and storage node. The function of computation nodes is the processing of jobs and applications. Computation nodes could be in a cluster, mainframe, high performance computer or a desktop which are capable of executing applications on many processors, maintaining registry (i.e. index of LOs) and policy of the regional grid and performing job request.

Storage nodes are the machines which are responsible for storing and providing data to other nodes inside the grid environment. The most common storage type is secondary storage using hard disk drives or other permanent storage media such as tape drives. Different types of file systems exist which will handle the storage and organisation processes for the data across the nodes of the regional grid network. Network File System (NFS), Distributed File System (DFS) and General Parallel File System (GPFS) are some popular network file systems.

In our case study both types of node represent institutions within the administrative region. An institution could have more nodes than others consequently an administrative region could have more nodes than others. The number of nodes in a particular administrative region depends on the size of that administrative region and the number of institutions. For instance, the Riyadh region has more nodes than the other 12 administrative regions in the KSA, this is because the Riyadh region had more than 57 universities and colleges and more than 183807 registered students in 2009 [28]. None of the other 12 administrative regions had more than this number of institutions and registered students.

3. *Data resources:* The data resources are distributed among regional grids and

### 4.3 The Computational Model of The RDADeLE System

---

are stored in storage nodes. The data we are concerned with is the LOs which are sharable and accessible under the policy of a regional grid. The LOs are stored in one node or more within a regional grid. The owner of the data has the right to maintain and determine the policy for the data. The policy on data resources could be read only, migrated, written and copied.

In our case study, data of each administrative region is distributed among nodes within that region. For instance, the Eastern province region has more than 30 universities and colleges and more than 76798 registered students, all data belonging to these universities and colleges are distributed within the Eastern province region. The same concept is applied for the other 12 administrative regions.

4. *Policy*: Policy is defined to be sets of rules, principles and practices explicated by one or more owners of a resource about how resources can be accessed and used. Policy is used to explain how security is implemented in an organisation and how an organisation manages, protects and distributes resources. Resource policies are stored in a separate media storage as an XML file. Each regional grid has its own policy which is based on the properties of that regional grid. Policy includes policy of regional grid data and applications.

In our case study there are differences in the policy of each administrative region. Although the main education policy in the KSA is a common policy some administrative regions have their own policy regarding their regional grid data. The difference in policies appear between countries, not within one country.

## 4.3 The Computational Model of The RDADeLE System

---

### 4.3.2 The Mechanisms

A mechanism manages and controls the RDADeLE system components using for instance, the following three mechanisms: *communication*, *timing* and *failure*.

1. *Communication Mechanism*: Communication between objects is needed in order for them to perform their activities. The aim of communication is to transport messages from one object to another in order to exchange and transmit information and data between those objects. Also, communication is used to help discover failures. Remote Procedure Calls (RPC) is the means of communication between sending and receiving objects. Communication may be either synchronous or asynchronous.

In our model, interactions between regional grids (containers) use Remote Method Invocation (RMI) and messaging services between the RDADeLE entities (i.e. agents) is performed using Message Transport Protocol (MTP). MTP manages all message exchange within and between platforms. Message-based asynchronous communication is the basic form of communication between agents. Using asynchronous communication in sending and receiving messages does not block synchronous communication. Asynchronous communication is useful in situations when it is possible for an object to retrieve replies later. The MTP transport FIPA-ACL messages between agents which contain a set of one or more message parameters. The parameters include performative (e.g. Request, Inform and Failure), sender, receiver and content.

2. *Timing Mechanism*: After a search request has been submitted, it starts looking for desired information from all over the regional grids. The search process runs in each regional grid. The regional grid node is responsible for the search request (i.e. the search process runs on regional grid node). A time is supposed to be

### 4.3 The Computational Model of The RDADeLE System

---

determine for each search request by each regional grid based on properties of that regional grid. If the search result is not obtained within the determined time the regional grid node informs the search sender that the allowed time for search has expired and gives the search sender the option to send the request again or ask the regional grid to perform another search request after killing the previous request.

In our case study, if a learner in the Riyadh administrative region sends a search request to all administrative regions including the Riyadh region, each administrative region allows a determined time for the search request. This is very important in raising the performance of the RDADeLE system.

3. *Failure Mechanism:* Failures in complex and distributed systems can be unpredictable. In these environments the probability of failure is high, for the following reasons:

- Complex and distributed systems combine an immense amount of hardware and software components.
- Complex and distributed systems could consist of heterogeneous entities, which can lead to failure when they interact.
- Complex and distributed systems are dynamic, with components constantly joining and leaving the system.

Fault tolerance is an essential function for complex and large environments in order to avoid the loss of computation, data and results. Fault tolerance mechanisms provide failure detection and failure handling. Keeping track of and connections between entities of a system plays a big role in the system coherence and control. The importance of this issue appears in complex and

#### 4.4 RDADeLE Architecture Components

---

large systems which deal with large amount of components and entities. Our RDADeLE model assumes that tracks or connections of the system components and entities may fail at any time. In our RDADeLE model, since agents control all components and entities of the e-learning system keeping track of all of them is the main role of fault tolerance. Information Service (IS) of the RDADeLE system (Main Container) keeps track of all agents and all “containers” (regional grids) which are connected to it. Replication of IS (Main Container) will guarantee the tracks and connections between RDADeLE agents and “containers” (regional grids).

In our case study, replication of IS (Main Container) will produce many replicated main containers to which all 13 administrative regions (containers) are supposed to be connected. The topology of connections between the Main Container and replicated main containers is ring topology.

The Main Container and replicated main containers will arrange themselves in a logical ring so that whenever one of them fails, the others will notice and act accordingly. Containers (regional grids) will then be able to connect to the platform through any of the active main containers; the different copies will evolve together using cross-notification.

Detection occurs using cross-notification between connected replicated containers when the Main Container or a replicated main container fails. Consequently, non-failed containers will arrange themselves in a logical ring.

#### 4.4 RDADeLE Architecture Components

The RDADeLE architecture was designed to achieve and fulfill its aims and objectives which include controlling and managing complex systems and system of systems.

#### 4.4 RDADeLE Architecture Components

Figure 4.3 shows an overview of the RDADeLE architecture which consists mainly of agents which live in the Information Service (IS) (i.e. container) to provide the internal structure. These agents are triggered when the state of the RDADeLE environment is changed or when a request is initiated by a user. The user in the architecture refers to a learner or a requester representative. The user sends requests through a portal (e.g. a web) which provides a user interface. The portal is a means to enable the user to send a request and receive a response to and from the system.

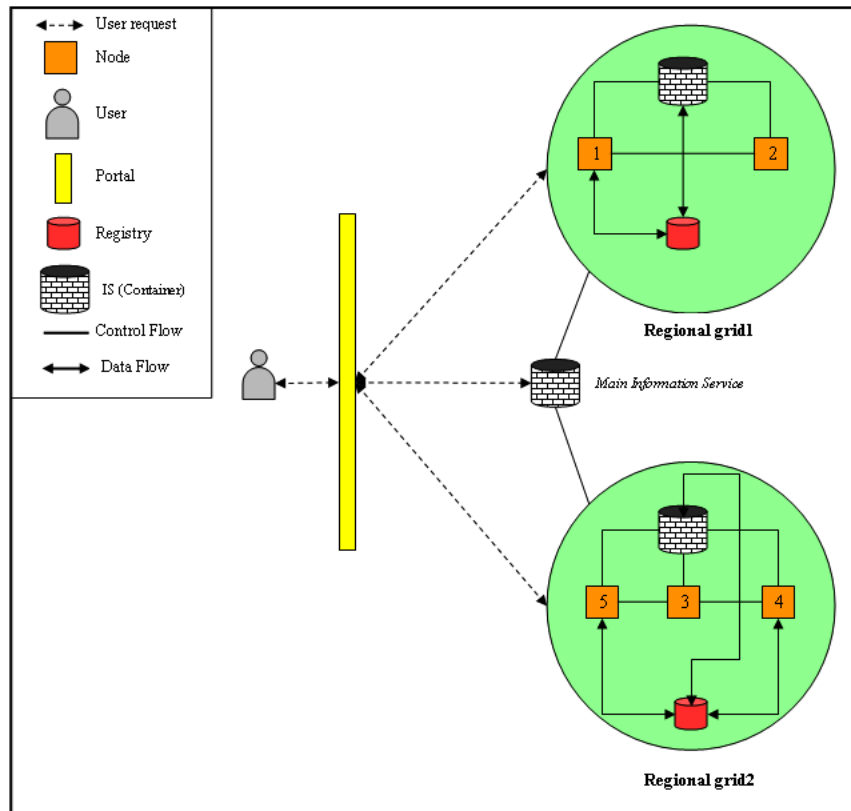


Figure 4.3: RDADeLE Overview

The brick cylinder shape represents the information services for the RDADeLE system. There is one Main Information Service (Main Container) and many information service (containers) which reside in each regional grid. Whereas the regular cylinder shape represents the registry of each regional grid. The registry of each

#### 4.4 RDADeLE Architecture Components

---

regional grid leads to the contents of desired LOs which are queried by requesters.

Dotted lines show the request path which travels to all regional grids connected to the RDADeLE system in order to search for LOs' information or grid services. The path also shows the results returning to the user. The dotted line between the portal and the Main Information Service represents the path used by the requester to obtain regional grids which are currently connected to the RDADeLE system. Using data grids and portals enables learners, employees and the general public to search for and collect information about LOs, courses, grid service, and degree plans from regional grids from all over the RDADeLE environment.

The square shapes represent nodes. Nodes represent a computation node or a data resource node and locations of servers which provide computation processes or repository of LOs' information/grid services. Each regional grid has one or more server which represents one or more academic institution or training centre.

The container of each regional grid works as an Information Service (IS) of all nodes within the regional grid. The solid lines (without arrows) show the control flow between components, for example, IS (container) in regional grid1 controls nodes labeled "1" and "2" within the regional grid. The solid lines with arrows in both sides show the data flow, for example, there is data flow between registry and data resource nodes labeled "1". On one hand, the role of one type of agent within each regional grid is both to help users to search and retrieve LOs' information/grid services and control data passing in/out from regional grids according to assigned constraints which are part of the properties of regional grids. On the other hand, the role of another type of agent is to update each regional registry.

The RDADeLE system is a multi agent system which is designed using a hierarchical structure of agents. We have adopted this approach in order to build dynamic and robust distributed data grids (i.e. regional grids). We consider that agents in the

## 4.4 RDADeLE Architecture Components

---

RDADeLE system are intelligent and autonomous. There are many reasons for this view. Firstly, intelligence is required in such multi-task systems. Secondly, intelligent agents support the execution of tasks in an optimal timescale. Autonomous agents can perform tasks in pursuit of a goal without direct supervision or control. These types of agents are necessary in the RDADeLE dynamic architecture.

### 4.4.1 Regional Grid Structure

Each regional grid represents one or more educational institutions and training centres [12] within an administrative region. Figure 4.4 has been derived from figure 4.3 which shows two regional grids, regional grid1 and regional grid2. Figure 4.4 shows the components of regional grid2 which consists of its registry, Regional Information Service (RIS) and three nodes, Node3, Node4 and Node5. Node4 and Node5 are data resource nodes, whereas Node3 is a computational node. Contents of LOs and grid services are stored in the data resource nodes, labeled Node4 and Node5, which are accessed and retrieved by requesters through the registry.

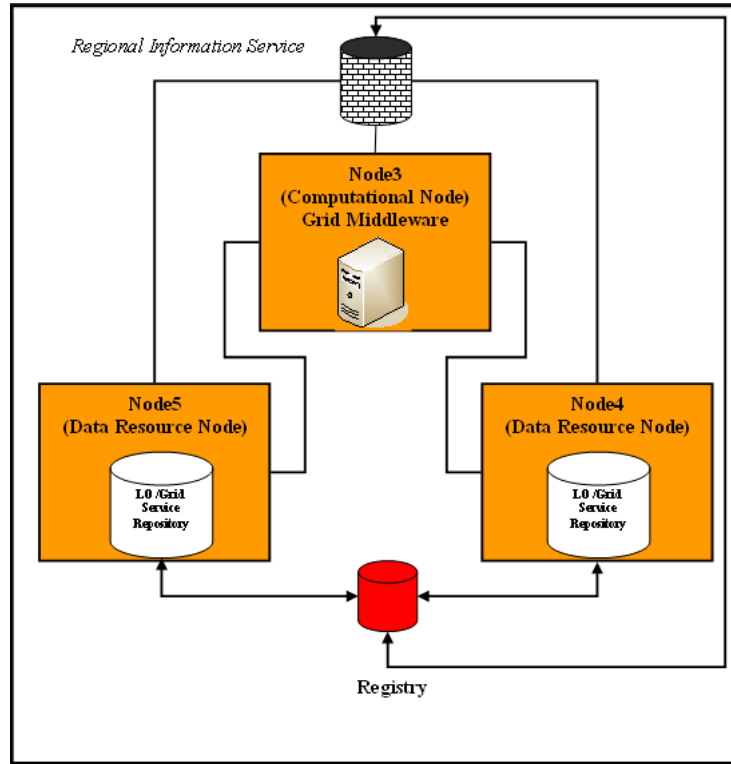
The RIS keeps track of and connections between the regional grid's components, i.e. Node3, Node4 and Node5. The solid lines (without arrows) connect the RIS with Node3, Node4 and Node5. The computational node, labeled Node3, has the grid middleware which helps in managing and organising the data grid within the regional grid. The solid lines (without arrows) connect Node3 with both Node4 and Node5. Maintaining the registry and searching for LOs' information and grid services is performed by agents which live in the RIS. The solid lines with arrows on both sides connect the RIS with the registry to represent communications between agents and registries.

The purpose of distributing the registries in the architecture is based on an approach which will be explained in this section. There are two approaches which could



#### 4.4 RDADeLE Architecture Components

---



**Figure 4.4:** Architecture of Regional Grid Node of RDADeLE

be adopted in the architecture. The first is to create a central global registry. The content of this global registry is the contents of all registries in each regional grid. Users can discover all LOs in all regional grids via the global registry. At the same time, registries in each regional grid are used to discover only that particular grid's LOs. Users can access global and regional registries to discover LOs. The second approach is based on assigning one regional registry for each regional grid. In this approach there is no central registry with which to discover LOs. Instead, regional registries are used to discover regional LOs.

In RDADeLE, components of each regional grid represent the active entities, while the regional grids support the infrastructure in the following ways:

- A member may dynamically connect and disconnect from RDADeLE.
- A regional grid is always supposed to be connected to RDADeLE.

## 4.5 Contents and Services Management

---

- Each regional grid has one registry which publishes all LOs connected to it.
- A registry of a particular regional grid is updated according to the number of LOs which are added to or removed from the regional grid.
- Each regional grid has its own constraints which are considered part of its properties. These constraints control information flow from and to regional grids.

There are many different reasons for dividing our architecture into regional grids (regional segments) which include the following; such a division helps to produce a sound structure which allows the designer a flexible approach in the design of constraint-based segments. The other reason is that regional grids could include one or more countries which have common properties (e.g. cultural factor). This will ease the way for any institution or training centre to connect to a regional grid with similar properties. Another reason for dividing our architecture is that it provides a flexible constraint setting for particular regional grids, which is useful in utilising each regional grid in particular, and the whole global grid in general. Division has been adopted in the RDADeLE system in order to reduce problems or congestion that may occur. Moreover, if for any reason a single regional grid is disconnected from the RDADeLE system it will not affect other regional grids in particular and the whole system in general.

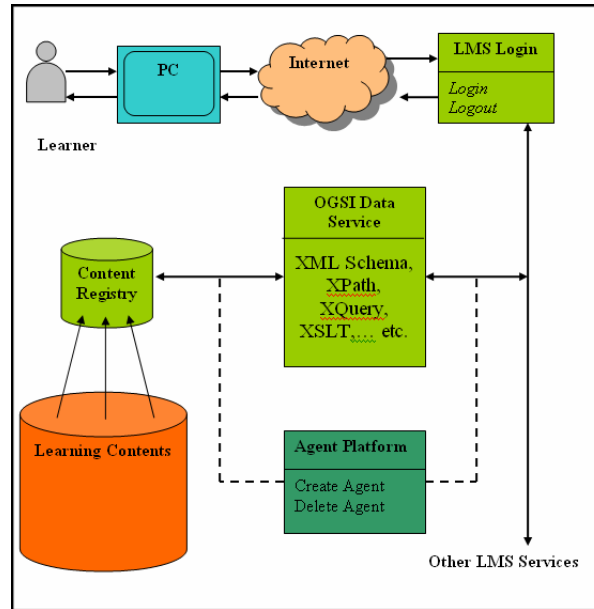
## 4.5 Contents and Services Management

The architecture encapsulates educational materials inside grid services which satisfies the demands of interoperability and reusability [118]. In a typical invocation of a web service, a client may use the UDDI Registry (Universal Description, Discovery and Integration), but in the architecture the client may use the registry for

## 4.5 Contents and Services Management

---

invocation of grid services. e-Learning content is published in the content registry in order to be accessible to requesters. The UDDI Protocol is used to find a server that hosts a grid service. A Learning Management System (LMS) is accessible via a web page to authenticate learners and search for LOs' information/grid services using agent assistance and grid middleware. Grid middleware includes Open Grid Services Architecture - Data Access and Integration (OGSA-DAI) and OGSI-complaint interfaces (OGSI- Data Service) which utilise programmes languages such as XML Schema, XPath, XQuery and XSLT (Figure 4.5).



**Figure 4.5:** Contents and Services Management

Datasets within regional grids are managed and controlled by both Data Grid Management Systems (DGMS) and agents. The agents' role in this context is to provide the means to simplify activities within regional grids. Functions that could be performed by agents include updating registries, searching for LOs' information/grid services, and authentication. Updating registries is indispensable in the e-learning environment in order to provide requesters with the correct and updated LOs' information. The update is performed in two situations. Firstly, when LOs' authors

## 4.6 Fault-Tolerance with Replicated Information Service

---

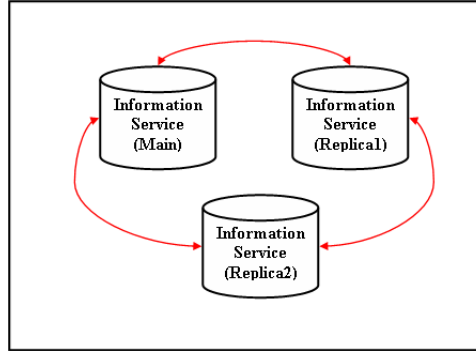
update the LOs' repository; and secondly, when the LOs are added to or removed from the regional grid. Another agent function is to be responsible for authentication of both the LMS and DGMS middleware via a single logon. All other LMS capabilities (discussion, chat, progress monitoring, accounting, course management and authoring) are considered to be web services[118].

## 4.6 Fault-Tolerance with Replicated Information Service

Fault tolerance is one of the main requirements that must be done when deploying real-world applications. Non-main containers of the JADE agent platform correspond to a regional grid whereas the Main Container corresponds to the information service of the RDADeLE system. The Main Container is used to house key platform services such as AMS and DF. This is a potential single point of failure that must be effectively managed to ensure the RDADeLE environment remains fully operational even in the event of Main Container failure. Consequentially, we make sure that all regional grids and their components in the RDADeLE system are fully operational even in the event of a failure of Information Service (IS). Replication of IS is an optimal technique which will be adopted. IS reflects information about all resources (regional grids and their components) in the RDADeLE system. Using this technique, it is possible to replicate any number of information services, which will arrange IS in a logical ring so that whenever one of them fails, the others will notice and act accordingly. The different copies will evolve together using cross-notification [19]. Figure 4.6 shows this technique. More detail will be provided in chapter 7, section 7.4.

## 4.7 Review of existing e-learning architecture towards RDADeLE Requirements

---



**Figure 4.6:** Fault-Tolerance With Replicated Information Service

## 4.7 Review of existing e-learning architecture towards RDADeLE Requirements

A number of existing e-learning architectures have been reviewed as possible candidate architectures which address the RDADeLE requirements in section 4.2. The RDADeLE architecture requirements are management, extensibility, flexibility and fault tolerance. Existing e-learning architectures do not include all these requirements; in particular, the following points highlight deficiencies of existing systems:

1. The architecture in [71] does not include collaboration with other learning management systems on other sites.
2. Learning Objects LOs are not considered as services.
3. Limitations usage between institutions and sites.
4. Data grid technology is not used.
5. Limitations usage of agents to mobile agents.

In contrast, the RDADeLE architecture satisfies the requirements by integrating data grid technology with agent technology. The first requirement is management of data resources. Management of data resources in the RDADeLE architecture is

## 4.8 Summary

---

carried out using data grid middleware without alterations so that all distributed data resources across a heterogeneous environment are managed (covered in chapter 2). Part of the management is monitoring all components of the RDADeLE architecture. The monitoring is carried out using agents which control and represent e-learning components such as regional grid, learners and nodes (covered in chapter 3).

The second requirement is the extensibility (scalability). Extensibility in data resources is carried out using data grid middleware without alterations to facilitate large-scale data-intensive computing and to provide large storage capacity so that all different types of data resources appear as a unified data resource which are shared between grid users (covered in chapter 2).

The third and fourth requirements, flexibility and fault tolerance respectively, are carried out using agent technology which has been covered in chapter 2. The conclusion is that the RDADeLE architecture does not have deficiencies which exist in existing e-learning architectures.

## 4.8 Summary

This chapter provided an overview of the architecture of the RDADeLE system. It started with requirement analysis which includes the previous e-learning architecture, motivation, and standards of learning objects and its metadata standards. Then, it described the computational model of the RDADeLE system guided by the KSA case study. Afterwards, it described architecture components and regional structure of the RDADeLE system. Finally, it described contents and services management and fault tolerance of the RDADeLE system.

# Chapter 5

## RDADeLE Agents' Specifications

### Objectives

---

- Review knowledge representation of the RDADeLE agents.
  - Present two different scenarios in the RDADeLE system.
  - Discuss the MAS approach in the RDADeLE system.
  - Present functions and descriptions of the RDADeLE agents.
- 

### 5.1 Introduction

The development of intelligent agent programs and expert systems is often labour intensive, time consuming and expensive, involving a number of knowledge formatting steps which include knowledge acquisition, knowledge analysis, system design and system implementation. During knowledge analysis and system design a knowledge level specification is produced. The knowledge level is an abstract knowledge system or agent specification which allows a knowledge engineer to describe the behaviour of the system under development in terms of its knowledge without making a commitment to its implementation architecture. Once the knowledge level representation is completed, it is transformed into an application.

This chapter presents RDADeLE agents relationships, which include communications and relationships between RDADeLE agents, the two RDADeLE scenarios, the MAS approach in the RDADeLE system and the architecture for each agent in the system.

## 5.2 The RDADeLE Knowledge Representation

We have adopted hierarchical knowledge representation in the RDADeLE system. This is because hierarchical knowledge representation is inheritable knowledge, which centres on relationships and shared attributes between kinds or classes of objects. These features are urgently needed in the RDADeLE system, since it is dynamic, large and complicated. Using hierarchical representation simplifies reasoning by limiting the number of distinguishing elements we have to deal with, reducing complexity, and thinking at a higher level of abstraction where possible [22]. Object-oriented programming languages such as C++ and Java provide a natural framework for representing knowledge as objects and for manipulating those objects. The hierarchical representation of knowledge is different from other kinds of representation, including procedural and relational representation.

## 5.3 RDADeLE Agents' Relationships

RDADeLE is a multi-agent system in which agents communicate with each other in order to complete and process their tasks. There are relations between these agents which give us a clear view of the RDADeLE system. This section begins by introducing the types of communication, then considers relationships among RDADeLE agents.

### 5.3.1 Communication

Communication is the essential means by which agents cooperate and coordinate in order to achieve goals. In a decentralized multi-agent system environment, there are different types of communication, which include Agent to Human, Agent to Agent, Agent to Non-Agent (object), and Agent to Environment.



### 5.3 RDADeLE Agents' Relationships

---

- Agent to Human: Agents may communicate with a human in different ways, such as through textual dialogs. Agents that communicate with humans are usually named interface or user agents. They serve users by accepting queries and returning the results to them.
- Agent to Agent: Communication between agents can be established through the communication components of their architecture. They exchange messages using predefined mechanisms and protocols.
- Agent to Non-Agent: agents may also be able to communicate with non-agents (e.g. databases, applications, and middleware) through their names and addresses.
- Agent to Environment: An agent may be able to communicate with an environment, including the type of operating system that it runs on.

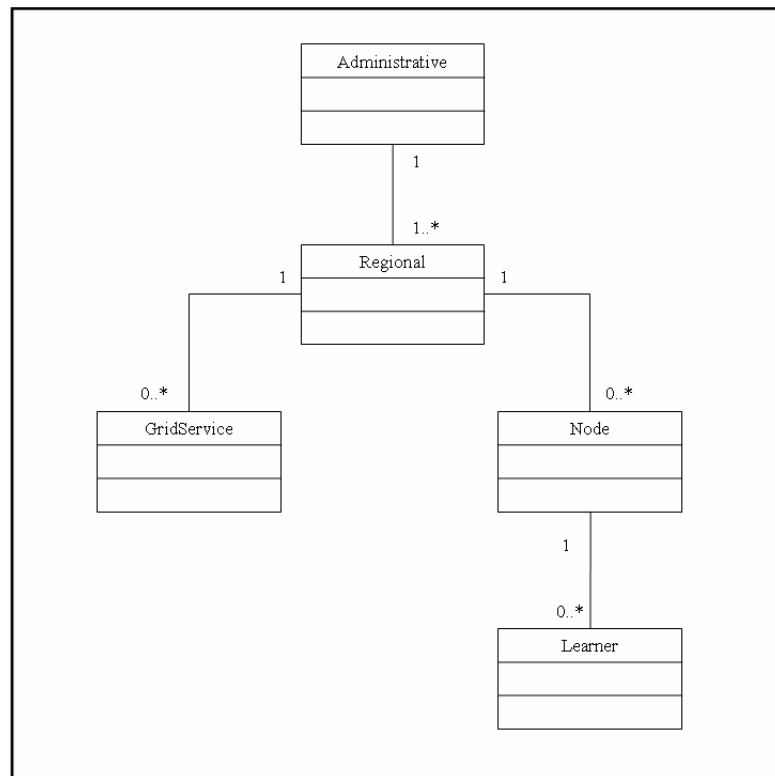
#### 5.3.2 Agent Relationships

One of the most obvious features of multi-agent systems is cooperation and collaboration between agents. In RDADeLE, agents control and administrate grid components, which include regional grids, users (learners and staff), grid services, information services and nodes. In order for the RDADeLE system to be dynamic, flexible and reliable, the agents communicate and cooperate among themselves. Figure 5.1 shows the relationships between RDADeLE agents.

Depending upon the architecture of RDADeLE, as shown in figure 5.1, the administrative agent class associates with one or more regional agents (regional grids). This means that at least one regional grid must exist to build the RDADeLE system. A regional agent relates to one or more node agents. There are some circumstances where there could be no node within a regional grid, for example in the case of no

### 5.3 RDADeLE Agents' Relationships

---



**Figure 5.1:** Class Diagram Association

## 5.4 RDADeLE Scenarios

---

institution being registered to a regional grid. In order to operate the RDADeLE system, there should be at least one learner to initiate requests. It is possible that there could be no learner related to a node, for example where no learner is registered to an institution. There is a one-to-many relationship between the regional agent class and the grid service class “GridService”. This means that there exists one or more grid services in the Directory Facilitator (DF) within a regional grid. Furthermore, it is possible that there could be no grid service related to a regional grid, where no grid service has been registered with the DF within a regional grid.

## 5.4 RDADeLE Scenarios

As we will see in detail in chapter 6, there are two techniques for composing and searching for LOs’ information and grid services: **the XML-based Registries Technique (XRT)** and **the Registered-based Services Technique (RST)**. The following subsections present two RDADeLE scenarios, each based on one of these techniques:

### 5.4.1 XRT Scenario

The first scenario corresponds to **the XRT**. The administrative agent initiates the creating of regional agents (regional grids) via the Agent Management System (AMS). AMS is an agent ready-made by the JADE platform which provides the naming service (i.e. ensures that each agent in the platform has a unique name) and represents the authority in the platform; for instance, it is possible to create/kill agents in remote containers by request to the AMS. This agent and the administrative agent are responsible for managing the platform and providing the white-page service. Once regional agents are created, they perceive the LOs’ information and regional policy by parsing the registry (which contains the LOs’ information) and policy files of their own

## 5.4 RDADeLE Scenarios

---

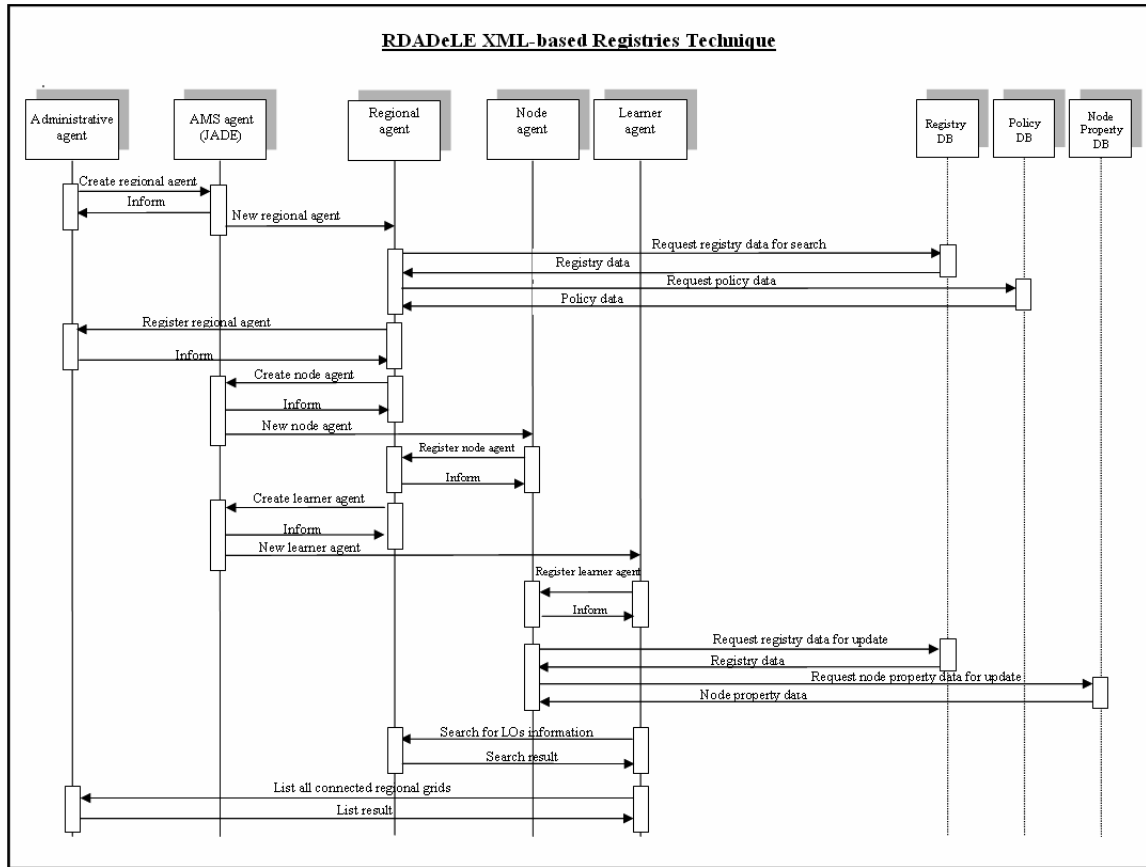
regional grid. Regional agents need to be authenticated in order to be registered for further needs. The authentication and registration are performed by administrative agents. Once regional agents are registered, they create their own components, which comprise node and learner agents. Each regional agent is responsible for registering nodes connected to it. The node agent is registered after its creation by the regional agent, while the learner agent is registered after its creation by the node agent. The aim of node agents is to maintain registry and node property files performed by authorised members of staff. Node agents perceive LOs' information and node property by parsing registry and node property files. One of the main tasks of a node agent is registering learners connected to it. Learner agents are initiated when a learner sends a request. These requests include a search for the LOs' information and a list of all connected regional grids. The search request is sent to all connected regional agents in order to retrieve required LOs' information from registries of all regional grids based on the regional policy. The list request is sent to administrative agents to list all connected (registered) regional grids. Figure 5.2 shows this first scenario.

### 5.4.2 RST Scenario

The second scenario corresponds to **the RST**, which differs from XRT in three ways:

- There are no registries for regional grids. Instead, there are grid services which are registered and kept in the DF. This is an agent ready-made by the JADE platform which provides a Yellow Pages service by means of which an agent can find other agents providing the services it requires in order to achieve its goals.
- The service agents (i.e. grid service agents) need to be registered with the DF.
- The search requests for grid services from learners are fulfilled by learner agents

## 5.4 RDADeLE Scenarios



**Figure 5.2:** XRT Scenario in RDADeLE

## 5.5 MAS-based RDADeLE

---

from the DF.

The administrative agent initiates the creation of regional agents (regional grids). Once regional agents are created, they perceive regional policy by parsing the policy files of their own regional grid. In a process similar to that under the first scenario, regional agents must to be authenticated in order to be registered for further needs. The authentication and registration are performed by administrative agents. Once regional agents are registered they create their own components, which comprise node, learner and service agents. Regional agents are responsible for registering nodes connected to them. The node agent is registered after its creation by a regional agent and the learner agent is registered after its creation by a node agent. The aim of the service agents is to describe the grid service by embedding the service type and property in order to be registered in the DF. One of the aims of node agents is to maintain node properties and this process is initiated by authorised members of staff. Node agents perceive node properties by parsing node property files. One of the main tasks of a node agent is registering learners connected to it. Learner agents are initiated when a learner sends a request which will include a search for grid services and a list of all connected regional grids. The search request is sent to the DF agent in order to retrieve the required grid services, based on the regional policy. The list request is sent to administrative agents to list all connected (registered) regional grids. Figure 5.3 shows this second scenario.

## 5.5 MAS-based RDADeLE

We intend to create a dynamic e-learning environment which depends on data grid and multi-agent technologies. In the architecture, services are distributed, since each regional data grid has its own registry. However, there could be many requests from

## 5.5 MAS-based RDADeLE

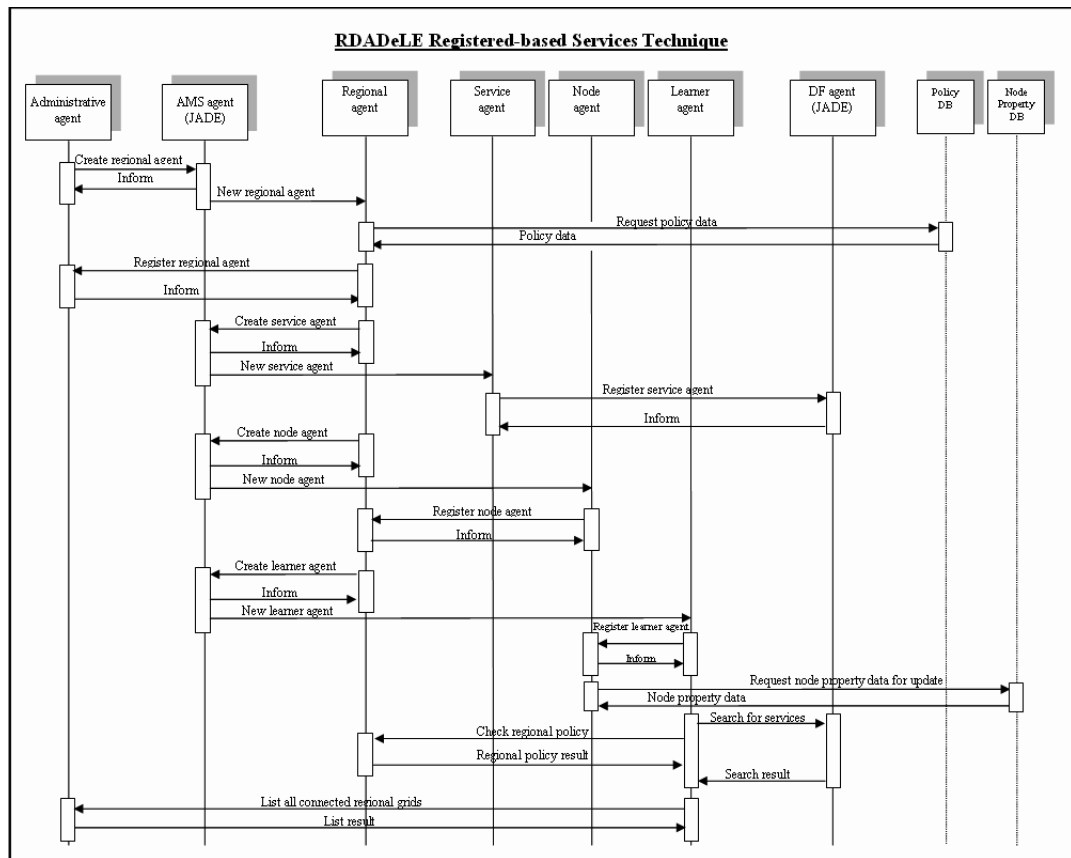


Figure 5.3: RST Scenario in RDADeLE

## 5.5 MAS-based RDADeLE

---

the portal which would cause a bottleneck in the services. Using distributed agents in this context will help in resolving part of this problem. Agents in the architecture have been designed to be autonomous. In order to make agents autonomous and have flexible behaviours, they have to be reactive and social. Reactivity means that an agent can perceive its environment and respond in a timely fashion. Agents perceive the learning environment through the requesters (i.e. learners) and service providers (institutions). Sociability means that agents are capable of interacting and communicating with other agents and humans. Agents in the architecture, in a social context, satisfy these features in interaction with requesters and in communication with each other.

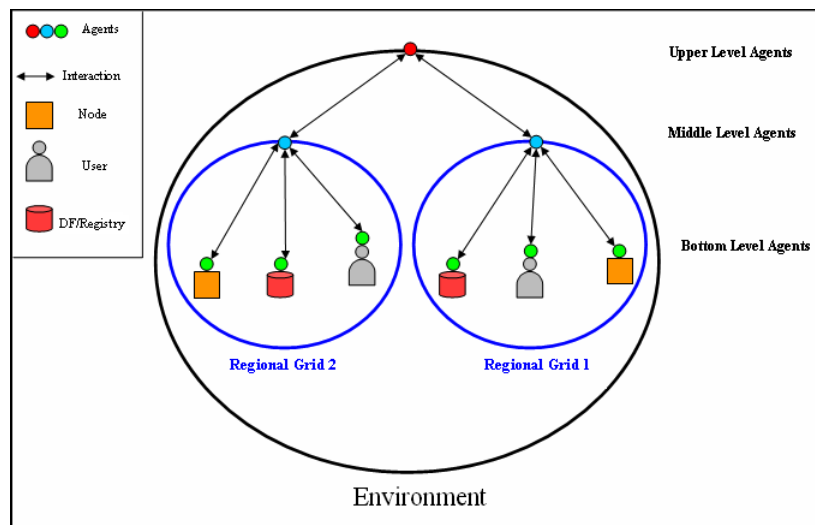
The primary concern of the MAS-based RDADeLE environment is the interaction of agents themselves and their relationships with their environment. These relationships are created in order to introduce a dynamic environment. Controlling and organising agents' behaviours is another aspect of producing a dynamic environment. The multi-agent environment plays a major role in relationships between regional grids themselves and between components within each regional grid. The main role of the multi-agent environment in regional grids is to build a dynamic, intelligent and collaborative environment [90].

The multi-agent structure used here is hierarchical (i.e. layered). This approach is ideal for solving large-scale, complex problems. Hierarchical structures have been adopted and studied in many fields of research including scientific computing and business processing. The basic idea of a hierarchical structure is that a complex system can be divided into subsystems; the overall behaviour of the system is determined by its subsystems, which perform sub-functions [134]. The Hierarchical layered MAS has three layers, the upper, middle and bottom levels, as shown in figure 5.4.

Agents need to receive behavioural instructions, whether from agents at higher



## 5.5 MAS-based RDADeLE



**Figure 5.4:** Hierarchical Agent Organisation of MAS-based RDADeLE

levels in the hierarchical structure or from other environmental components (e.g. learners), and receive support from agents at lower-levels to perform their tasks. In the model, *administrative agents* correspond to the upper-level agents, responsible for controlling and managing other agents at lower levels, who in turn are responsible for regional grid communication. *Regional grids* correspond to middle level agents, which are responsible for supporting activities within regional grids. *Regional grid components* (i.e. nodes, learners and grid services) correspond to bottom level agents, which are considered to be sensors of the environment.

Agent architecture is essentially a map of the internals of an agent. It includes the agent's data structure, the operations that may be performed on such data structures and control of the flow between them. There are many agent architectures which have been adopted in numerous applications. These include logic-based architecture, reactive architecture, belief-desire-intention architecture (BDI), layered architecture and deliberative architecture. Subsumption architecture is arguably the best-known reactive agent architecture [148], in which agent decision making is achieved through the interaction of a number of behaviours. All agents in the RDADeLE environment

## 5.6 Functions and Descriptions of RDADeLE Agents

---

are subsumption agents. This type of agent has a behaviour-based architecture which decomposes complicated intelligent behaviours into many simple behaviours, which in turn are organised into layers.

## 5.6 Functions and Descriptions of RDADeLE Agents

We have adopted the Prometheus methodology to construct the RDADeLE system. The Prometheus methodology consists of three phases: system specification, architectural design and detailed design phases. As stated above, there are agents at three levels. These are upper-level, middle-level and bottom-level agents [11]. The upper and middle levels consist of one type each: *administrative* and *regional* agents respectively. The bottom-level consists of three types: *node agents*, *service agents* and *learner agents*. Hence the RDADeLE system comprises five distinct agent types:

- Administrative Agent.
- Regional Agent.
- Node Agent.
- Service Agent.
- Learner Agent.

In the model, the upper-level (*administrative*) are responsible for monitoring and controlling the middle-level (*regional*) agents. These are intermediate agents which in turn are responsible for monitoring and controlling the behaviour of a regional zone whose components include a number of bottom-level agents *node agents*, *service agents* and *learner agents*. This means that the primary role of these agents is monitoring and controlling behaviours throughout the whole environment.

## 5.6 Functions and Descriptions of RDADeLE Agents

---

In a regional grid, possible requests can be represented as a set

$$R = \{r0, r1, r2, \dots r_k\}. \quad (5.1)$$

Once the learner agent perceives the request, it will send the request to other agents (most requests go to regional agents as search requests) to be fulfilled. Accordingly, these respond to each request as an action. Agent capabilities can be represented by a set of actions:

$$A = \{a0, a1, a2, \dots a_l\}. \quad (5.2)$$

The set of regional agents (i.e. regional grids) is as follows:

$$RG = \{rg0, rg1, rg2, \dots rg_m\}. \quad (5.3)$$

For each learner request, there exist constraints which form a set of constraints:

$$C = \{c0, c1, c2, \dots c_p\}. \quad (5.4)$$

This process of agent action generation can be represented as a function:

$$action : R \times C \longrightarrow A \quad (or \quad a_n = action(r_n, c_n)). \quad (5.5)$$

This process of agent actions in responding to requests is distributed among regional agents according to the following function:

$$\frac{\sum_{i=0}^n a_n = action(r_n, c_n)}{m - 1}. \quad (5.6)$$

### 5.6.1 Administrative Agents

Administrative agents, with AMS and DF agents, provide an information service for the RDADeLE system. The administrative agent is the first agent created to build

## 5.6 Functions and Descriptions of RDADeLE Agents

---

and monitor the RDADeLE system. It creates regional agents (regional grids) to build the RDADeLE system, while regional agents in turn build its components (i.e. nodes, learners and grid services). The administrative agent supplies authentication services for regional grids to register with the RDADeLE system. Meanwhile, the administrative agent controls and manages agents at the lower level (i.e. regional agents) and collects and preserves information about the system, including registered regional grids and the capacity of the RDADeLE system. Furthermore, it is responsible for registering regional grids with the RDADeLE system. This information helps in searching for LOs' information and grid services.

*The administrative agent* is described in a tuple, as follows:

$$AA = \langle cr, e \rangle \quad (5.7)$$

where:

- $(cr)$  is a control request and
- $(e)$  is an entity.

The control request  $(cr)$  includes the creating, termination, (de)registerion and authentication of regional agents. The entity  $(e)$  is a regional agent. The administrative agent has some functions, which are summarised as follows:

1. Monitoring the RDADeLE system.
2. Creating regional agents (regional grids).
3. Authenticating the regional grid that connects it to the RDADeLE system.
4. Registering/deregistering the regional grid on request from the regional agent.
5. Listing all registered regional grids upon request from learners.

## 5.6 Functions and Descriptions of RDADeLE Agents

---

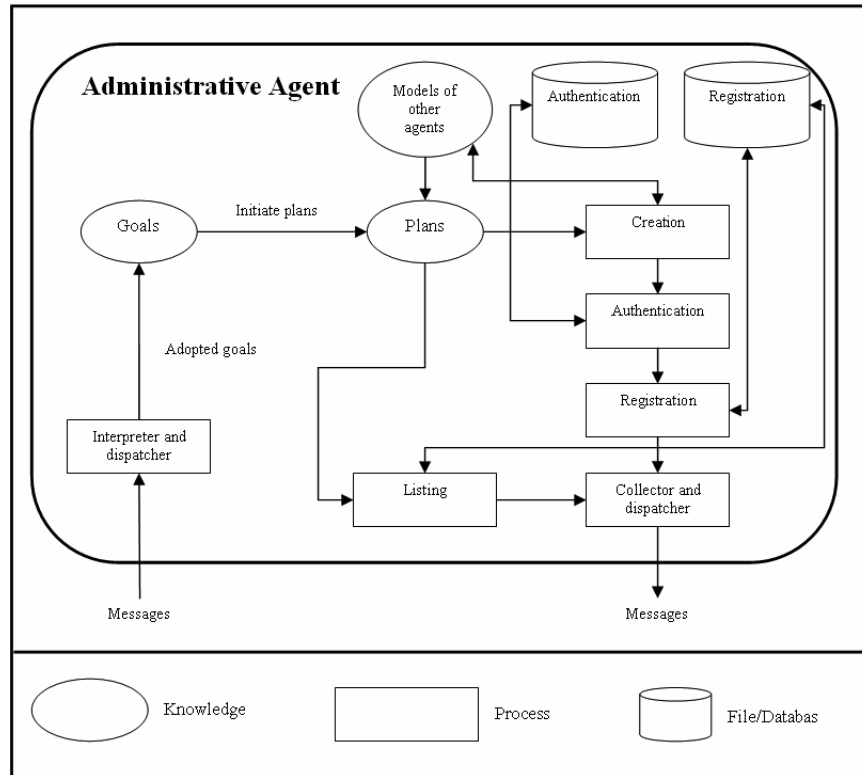
### 5.6.1.1 Administrative Agent Architecture

Figure 5.5 shows the administrative agent architecture. The agent has six main components, which are:

1. Interpreter and dispatcher.
2. Creation.
3. Authentication.
4. Registration.
5. Listing.
6. Collector and dispatcher.

The first component is the interpreter and dispatcher, which receives incoming messages to the administrative agent from outside. The messages are interpreted and dispatched to the next component. The aim of the message will be identified as goals, then translated into plans to fulfil these goals. The plans here are agent behaviours and actions. The behaviours are constructed in cooperation with other agents if needed. The first action accomplished by the administrative agent is to create regional agents (regional grids). The creation behaviour is performed using the creation component. The newly created regional agents must be authenticated in order to be registered (connected) to the RDADeLE system. The authentication behaviour will be executed through the authentication component. The registration behaviour will take place in order to register the regional agent. The registration process, which includes both behaviour registration and deregistration, is performed through the registration component. The listing component is responsible for listing all registered regional grids upon request from learners. The last component

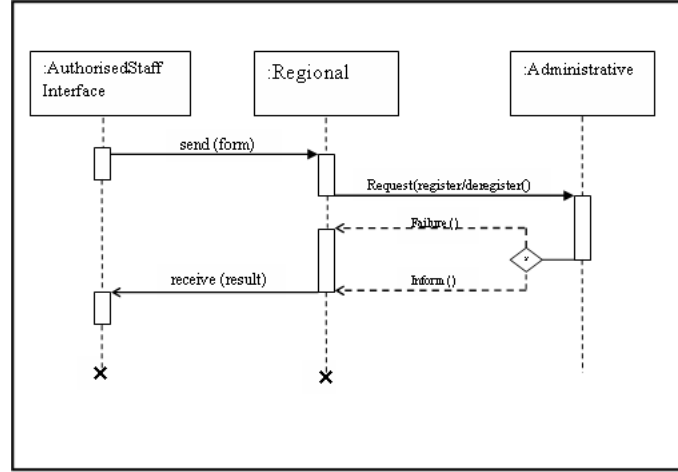
## 5.6 Functions and Descriptions of RDADeLE Agents



#### 5.6.1.2 Administrative Agent Design Model

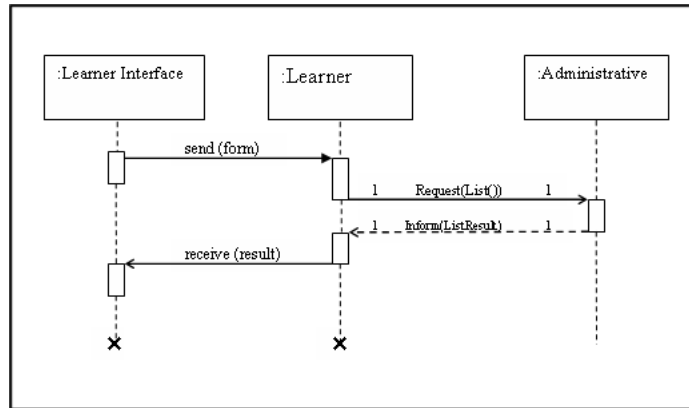
Figure 5.7 shows the design model of the listing request using AUML. The administrative agent fulfils the request to list all registered regional grids (request from

## 5.6 Functions and Descriptions of RDADeLE Agents



**Figure 5.6:** Registration of Regional Grid

learners).



**Figure 5.7:** Listing of Registered Regional Grids

### 5.6.1.3 Administrative Agent Algorithms

Monitoring the RDADeLE system is another task of the administrative agent. The following two algorithms, each corresponding to a behaviour, explain how the administrative agent monitors the RDADeLE system:

First, in Algorithm 1, “*TickerBehaviour*” behaviour is performed periodically every 20 seconds to inform the RDADeLE system’s administrator of the current situation of the system. *numberOfTotalAgents* is a HashMap of String and Integer which

## 5.6 Functions and Descriptions of RDADeLE Agents

---

contains regional names and the total numbers of their components. *maxNumberOfAgents* is the maximum number of agents that can be accommodated in one regional grid, which is 3000.

---

**Algorithm 1** TickerBehaviour Algorithm in the Administrative Agent

---

```
for each connected regional grid do
  if  $numberOfTotalAgents \geq (maxNumberOfAgents - (maxNumberOfAgents * 0.1))$  then
    Print “The current capacity of the regional grid is 90% of its full capacity”
  end if
end for
```

---

The second behaviour, in Algorithm 2, is “*CyclicBehaviour*”, which is performed periodically to receive messages from other agents. The message type *INFORM\_NUMBER\_AGENTS* determines the type of the message and informs the administrative agent of the current number of agents which live in a particular regional grid.

---

**Algorithm 2** CyclicBehaviour Algorithm in the Administrative Agent

---

```
msg  $\leftarrow$  receive()
type  $\leftarrow$  msg.getUserDefinedParameter(MSG_TYPE)
if type = INFORM_NUMBER_AGENTS then
  content  $\leftarrow$  msg.getContent()
  sender  $\leftarrow$  msg.getSender()
  numberOfTotalAgents  $\leftarrow$  (sender, content)
end if
```

---

### 5.6.2 Regional Agents

Since we have two techniques for composing and searching for LOs’ information and grid services, i.e. **XRT** and **RST**, we have two kinds of regional agent, providing the XRT regional agent and RST regional agent respectively. The former is responsible for dealing with the registry of LOs’ information, while RST regional agent is responsible for dealing with registered grid services. Both agents have the same architecture and behaviour, except that the RST regional agent does not have a search



## 5.6 Functions and Descriptions of RDADeLE Agents

---

component. These two agent types control the components of regional grids and work as regional grid facilitators. Registering nodes connected to the regional grid is performed by regional agents. Regional grids include registries of LOs' information and grid services, which is delivered to learners upon search request. The search results are delivered to learners according to the policies (constraints) which were set by authorised member of staff of each regional grid.

*The regional agent can again be described in a tuple, as follows:*

$$RA = \langle rgn, cr, e, se, c \rangle \quad (5.8)$$

where:

- (*rgn*) is the regional grid name,
- (*cr*) is the control request,
- (*e*) is the entity,
- (*se*) is the search parameter and
- (*c*) is the constraint.

The regional grid name (*rgn*) is a unique name. The control request (*cr*) includes creating, terminating, authenticating and registering/deregistering regional grid components. The entity (*e*) is the agent to be controlled (e.g. node, learner agents). The search parameter (*se*) includes the keywords used in searching for LOs' information. The constraint (*c*) is the policy of a regional grid which is applied in delivering this information. The regional agent has a number of functions, which can be summarised as follows:

1. Creating components of the regional grid (i.e. node, learner, and service agents).

## 5.6 Functions and Descriptions of RDADeLE Agents

---

2. Registering nodes upon request from the node agent.
3. Parsing the registry file (which contains the metadata of LOs' information in the current regional grid).
4. Parsing the policy file.
5. Searching for desired LOs' information.

### 5.6.2.1 Regional Agent Architecture

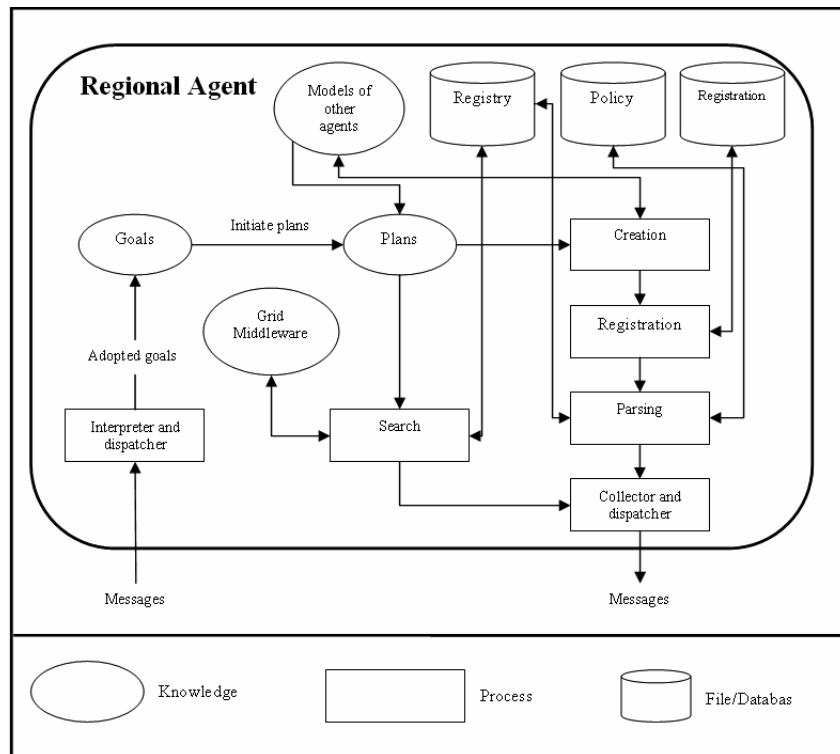
Figure 5.8 shows the regional agent architecture. There are six main components of the agent:

1. Interpreter and dispatcher.
2. Creation.
3. Registration.
4. Parsing.
5. Search.
6. Collector and dispatcher.

The first component is the interpreter and dispatcher, which receives incoming messages to the regional agent from outside. These are interpreted and dispatched to the next component. The aim of each message will be identified as goals and translated into plans to fulfil these goals. The plans here are agent behaviours and actions. The behaviours are constructed in cooperation with other agents if needed. The first action accomplished by the regional agent is the creation of its components, which include node, learner and service agents. The creation behaviour is performed

## 5.6 Functions and Descriptions of RDADeLE Agents

using the creation component. The registration behaviour will take place after the creation process in order to register the newly created node agent. The registration is performed through the registration component. The next operation is parsing both registry of LOs' information - if the technique used is **XRT** - and the policy of the current regional grid. The parsing process is performed via the parsing component. If the behaviour is a search query - if the technique used is **XRT** - the behaviour will be executed through the search component. For example, the message from the learner is a search request which will be transformed into a search goal. The aim of the search request is to search for LOs' information, which is translated into a plan. Subsequently, the plan is executed as a behaviour. The search process contacts the registry to look for the LOs' information. The last component is the collector and dispatcher, which is responsible for collecting the results of other all components, then forming the appropriate message to be dispatched outside the agent.

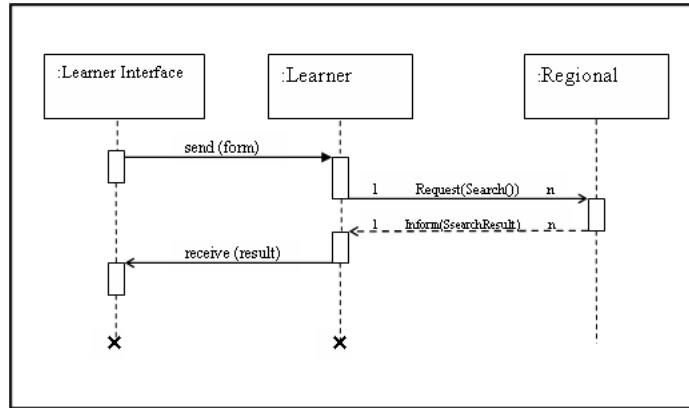


**Figure 5.8:** Regional Agent Architecture

## 5.6 Functions and Descriptions of RDADeLE Agents

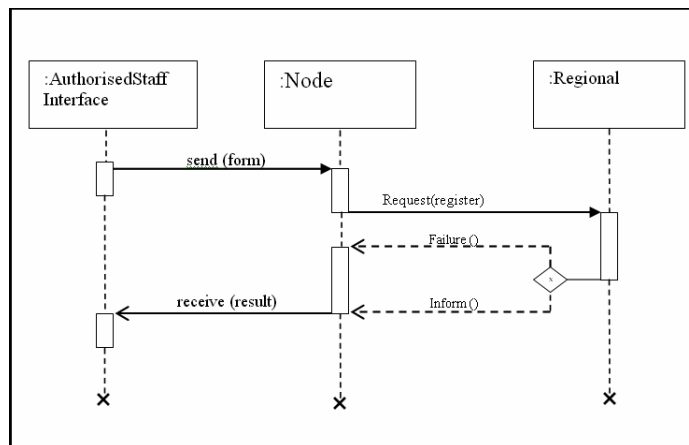
### 5.6.2.2 Regional Agent Design Model

A cyclic behaviour is added to each registered regional agent -if **XRT** is used- in order to enable and help learners to search for LOs' information using keywords. Figure 5.9 shows the design model of the search request for LOs' information using AUML.



**Figure 5.9:** Search Request for LOs' Information

The node agent sends a registration request to the regional agent in order to connect itself to the RDADeLE system and receives acknowledgment messages from it. Figure 5.10 shows the design model of the registration request using AUML.



**Figure 5.10:** Registration of Node

## 5.6 Functions and Descriptions of RDADeLE Agents

---

### 5.6.2.3 Regional Agent Algorithms

The regional agent plays a role in monitoring the RDADeLE system in cooperation with the administrative agent. The following two algorithms, each corresponding to a behaviours, explain how the regional agent cooperates with the administrative agent to monitor the RDADeLE system:

First, in Algorithm 3, the “*TickerBehaviour*” behaviour is performed periodically every 10 seconds to request current numbers of components connected to each node connected to it. *numberOfLearners* is a HashMap of String and Integer which contains node names and total numbers of their components. This behaviour calculates the number of components of the regional grid, *totalAgents*, then sends it to the administrative agent. Components of the regional grid include nodes, learners and services.

---

**Algorithm 3** TickerBehaviour Algorithm in the Regional Agent

---

```
totalNumberOfLearners ← 0
for each connected node do
  msg ← ACLMessage.REQUEST
  msg.addReceiver(node)
  msg.addUserDefinedParameter(REQUEST_GET_LEARNERS)
  send(msg)
  totalNumberOfLearners ← (totalNumberOfLearners +
    numberOfLearners.get(node))
end for
totalAgents ← (totalNumberOfLearners + nodeAgents.size() +
  serviceAgents.size())
msg ← ACLMessage.INFORM
msg.addReceiver(administrative)
msg.addUserDefinedParameter(INFORM_NUMBER_AGENTS)
msg.setContent(totalAgents)
send(msg)
```

---

The second behaviour, in Algorithm 4, is “*CyclicBehaviour*”, which is performed periodically to receive messages from other agents. The message type *INFORM\_GET\_LEARNER* determines the type of message which informs the regional agent of the current number of agents which live in a particular node.

## 5.6 Functions and Descriptions of RDADeLE Agents

---

---

**Algorithm 4** CyclicBehaviour Algorithm in the Regional Agent

---

```
msg ← receive()  
type ← msg.getUserDefinedParameter(MSG_TYPE)  
if type = INFORM_GET_LEARNER then  
    content ← msg.getContent()  
    sender ← msg.getSender()  
    numberOfLearners ← (sender, content)  
end if
```

---

The RDADeLE system reduces its limits for monitoring and controlling the whole environment which is termed the scalability. The scalability feature of the RDADeLE system is performed via cooperation between agents, of which one is the regional agent. The scalability of the system is performed by dividing the regional grid into sub-regional grids. The RDADeLE system checks the capacity of each regional grid periodically and checks the capacity before adding a new node to a regional grid. The regional agent checks its capacity before adding a new node to it. If the capacity of a regional grid has reached its limit, it requests the administrative agent to create a new sub-regional grid to accommodate new nodes, learners and grid services. Algorithm 5 is the reaction of the regional agent when it wants to add a new node and there is no available space, since it has reached its limit, considering that *nodeAgents* is a HashMap of String and Agent ID (AID), which contains current node names and addresses:

### 5.6.3 Node Agents

A node agent controls a node in a single regional grid, which represents a member of staff of an institution authorised to maintain registries and node properties. The registering of learners connected to a node is performed by the node agent.

*The node agent* can also be described in a tuple, as follows:

$$NA = \langle nn, rg, pr \rangle \quad (5.9)$$

## 5.6 Functions and Descriptions of RDADeLE Agents

---

---

**Algorithm 5** Extension Algorithm in the Regional Agent

---

```
1: learnersOfAllNodes  $\leftarrow$  0
2: regionalCapacity  $\leftarrow$  0
3: for each connected node do
4:   learnersOfAllNodes  $\leftarrow$  (learnersOfAllNodes +
     numberOfLearners.get(node))
5: end for
6: regionalCapacity  $\leftarrow$  (learnersOfAllNodes + nodeAgents.size())
7: if regionalCapacity < maxNumberOfAgents then
8:   Call createNewAgent(nodeName, "com.salehsz.agent.Node", arguments)
9: else
10:  Print "The capacity of regional grid is full."
11:  Print "New sub-regioanl grid will be created and the node will
    be attached to it."
12:  Call Administrative.createRegional(0,1,0,newRegionalName)
13: end if
```

---

where:

- (*nn*) is the node name (i.e. authorised member of staff),
- (*rg*) is the registry of a regional grid and
- (*pr*) is the properties of the node.

The parameters of the equation are (*nn*), which is the node name, (*rg*), which is the registry of a the regional grid, and (*pr*), which is the properties of the node, including node type, operating system, CPU, memory and available times. The node agent has a number of functions which are summarised as follows:

1. Parsing registry and node property files.
2. Maintaining registry and node property files by an authorised member staff.

### 5.6.3.1 Node Agent Architecture

Figure 5.11 shows the node agent architecture. The agent has five main components:

1. Interpreter and dispatcher.

## 5.6 Functions and Descriptions of RDADeLE Agents

---

2. Registration.
3. Parsing.
4. Maintaining.
5. Collector and dispatcher.

The first component is the interpreter and dispatcher, which receives incoming messages to the node agent from outside. These are interpreted and dispatched to the next component. The aim of each message will be identified as goals and translated into plans to fulfil these goals. The plans here are the agent behaviours and actions of the agent. The behaviours are constructed in cooperation with other agents if needed. The registration behaviour takes place in order to register the newly created learner agent and is performed through the registration component. The role of the parsing component in the node agent architecture is to parse both registry and node property files. If the behaviour is maintaining request, it will be executed through the maintaining component. Maintaining includes maintaining the registry file, grid services information and node property files. The last component is the collector and dispatcher, which is responsible for collecting the results of all other components and forming the appropriate message to be dispatched outside the agent.

### 5.6.3.2 Node Agent Design Model

A learner agent sends a registration request to the node agent in order to connect itself to the RDADeLE system and receives an acknowledgment messages from it. Figure 5.12 shows the design model of the registration request using AUML.



## 5.6 Functions and Descriptions of RDADeLE Agents

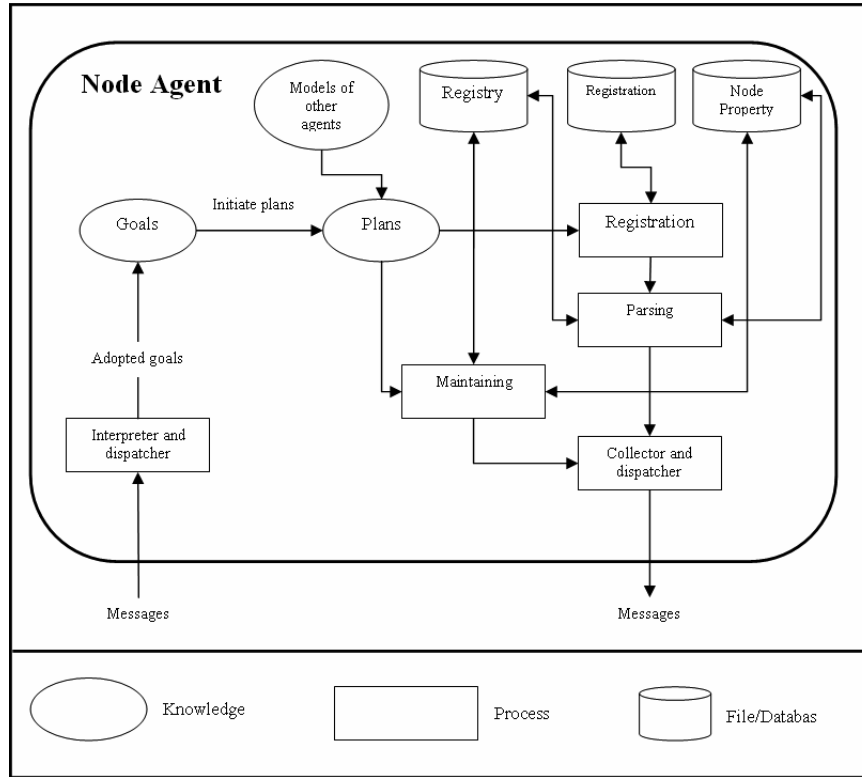


Figure 5.11: Node Agent Architecture

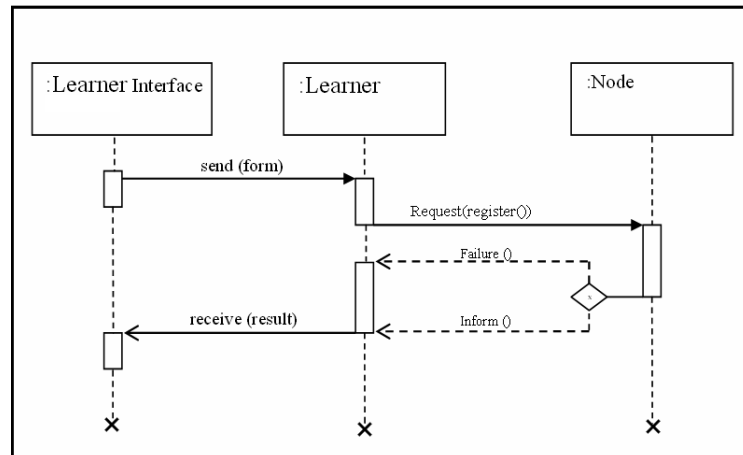


Figure 5.12: Registration of Learner

## 5.6 Functions and Descriptions of RDADeLE Agents

---

### 5.6.4 Service Agents

The service agent exists only in the **RST** scenario. This agent controls the grid service in each regional grid to be discovered by learners.

*The service agent* can also be described in a tuple, as follows:

$$SA = \langle sn, des, own, st \rangle \quad (5.10)$$

where:

- $(sn)$  is the service name,
- $(des)$  is the service description
- $(own)$  is the service owner and
- $(st)$  is the service state.

The parameters of the equation are  $(sn)$  which is the service name and is denoted as the unique name, the service description  $(des)$  which includes service type and properties, the service owner  $(own)$ , which is the regional grid to which the service belongs, and the state of the service  $(st)$ , which can take either of the following values: registered or deregistered. The aim of the service agent is to register the grid service with its description with the DF, for discovery and maintainance purposes.

#### 5.6.4.1 Service Agent Architecture

Figure 5.13 shows the architecture of the service agent, which has three main components:

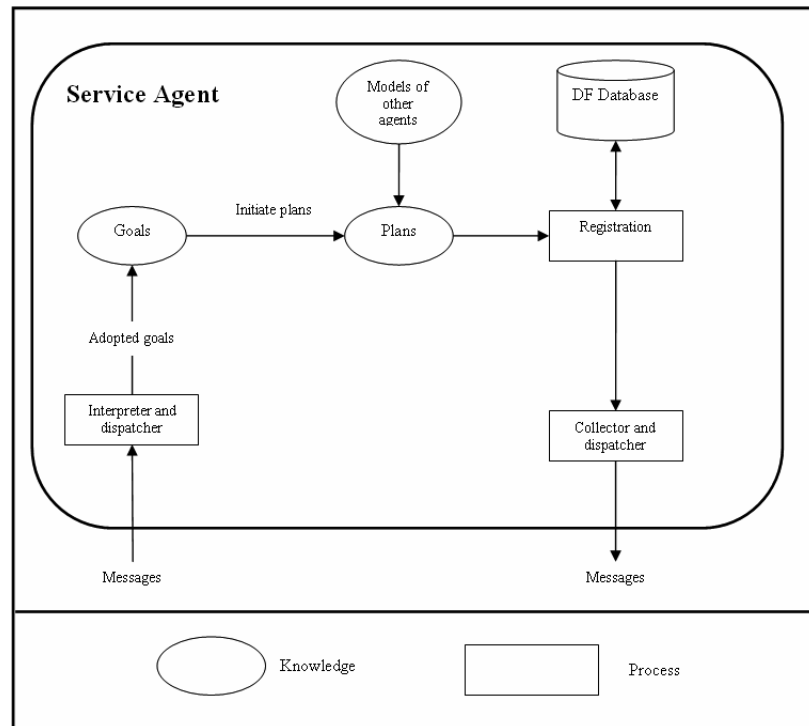
1. Interpreter and dispatcher.
2. Registration.

## 5.6 Functions and Descriptions of RDADeLE Agents

---

### 3. Collector and dispatcher.

The interpreter and dispatcher receives incoming messages to the node agent from outside. These are interpreted and dispatched to the next component. The aim of each message is identified as goals and translated into plans to fulfil these goals. The plans here are agent behaviours and actions. The behaviours are constructed in cooperation with other agents if needed. The behaviour is the registration request, which will be executed through the registration component. The registration component registers grid services and their descriptions with the DF, which may be expanded using a database. Grid service description includes type, name, owner and properties. The last component is the collector and dispatcher, which is responsible for collecting the results of all other components and forming the appropriate message to be dispatched outside the agent.



**Figure 5.13:** Service Agent Architecture

## 5.6 Functions and Descriptions of RDADeLE Agents

### 5.6.4.2 Service Agent Design Model

The aim of the agent is to register a grid service with the DF. First, an authorised staff member sends a request to register a new grid service. The service agent receives the request, then registers the service after determining its description. The agent then sends the service description to the DF agent to be kept for future use. An inform message then sent to the authorised staff member to acknowledge the registration result. Figure 5.14 shows the design model of the request to register a grid service using AUML.

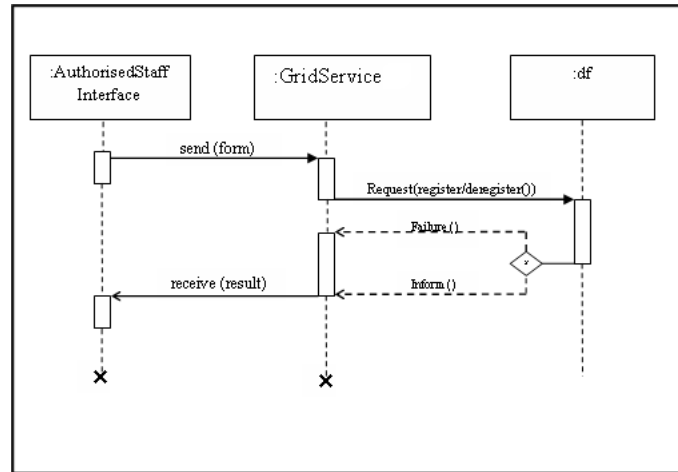


Figure 5.14: Registration of Grid Service

### 5.6.5 Learner Agents

A learner agent controls a learner and works as a learner facilitator, sending a queries about LOs' information, registered grid services and lists of connected regional grids.

*Learner agents* can also be described in a tuple, as follows:

$$LA = \langle ln, r \rangle \quad (5.11)$$

where:

## 5.6 Functions and Descriptions of RDADeLE Agents

---

- $(ln)$  is the learner's name and
- $(r)$  is the request.

Each learner name  $(ln)$  is a unique name. Requests from learners  $(r)$  include searching for the LOs' information, grid services and listing registered regional grids.

The functions of the learner agent can be summarised as follows:

1. Sending search requests for LOs' information to all registries of regional grids, which applies to **the XRT**.
2. Sending search requests for registered grid services, which applies to **the RST**.
3. Sending list requests to the administrative agent to list all registered regional grids.

### 5.6.5.1 Learner Agent Architecture

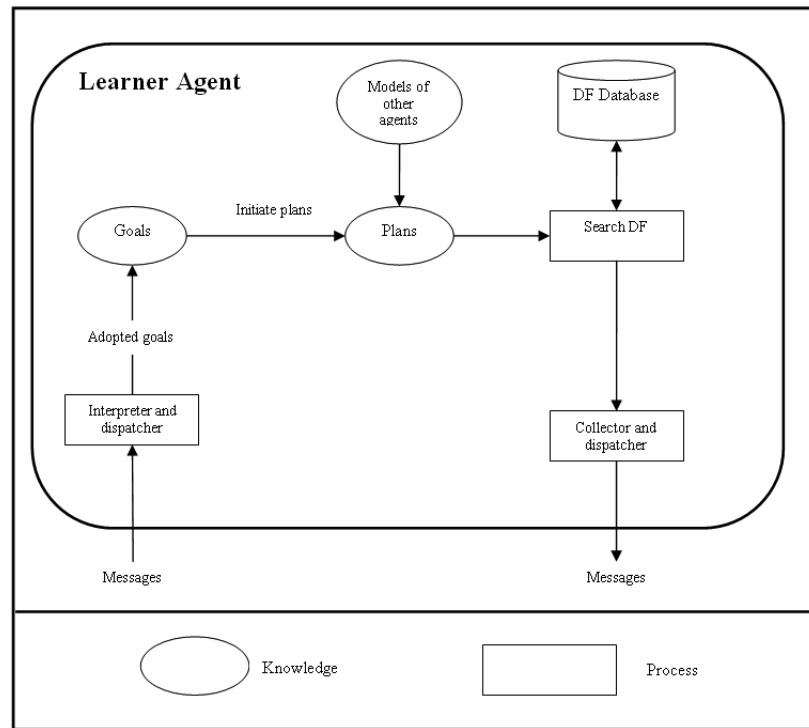
Figure 5.15 shows the architecture of the learner agent, which has three main components:

1. Interpreter and dispatcher.
2. Search for grid services in DF.
3. Collector and dispatcher.

This learner agent architecture description applies to **the RST**. The first component is the interpreter and dispatcher which receives incoming messages to the node agent from outside. These are interpreted and dispatched to the next component. The aim of each message is identified as goals and translated into plans to fulfil these goals. The plans here are agent behaviours and actions. The behaviours

## 5.6 Functions and Descriptions of RDADeLE Agents

are constructed in cooperation with other agents if needed. The behaviour, which is searching for registered grid services in the DF, will be executed through the Search DF component. In order to perform the search request, this component requires certain information, including agent description, service description, type of service and properties of the service, all of which is encapsulated in a template. The last component is the collector and dispatcher, which is responsible for collecting the results of all other components and forming the appropriate message to be dispatched outside the agent. Learner agent architecture in the **the (XRT)** does not have a search component, which exists in the regional agent architecture, as stated above.



**Figure 5.15:** Learner Agent Architecture

### 5.6.5.2 Learner Agent Design Model

The aim of the agent is to search for registered grid services in the DF. Figure 5.16 shows the design model of the search request using AUML. The agent has a

## 5.7 Review of RDADeLE agents' specifications towards RDADeLE Requirements

behaviour type named *CyclicBehaviour*, whereby it receives incoming messages and distinguishes between them by message type. The first message is the response to the search request and the second is the response listing all registered regional grids. Figure 5.16 shows the design model of the search request for grid services using AUML.

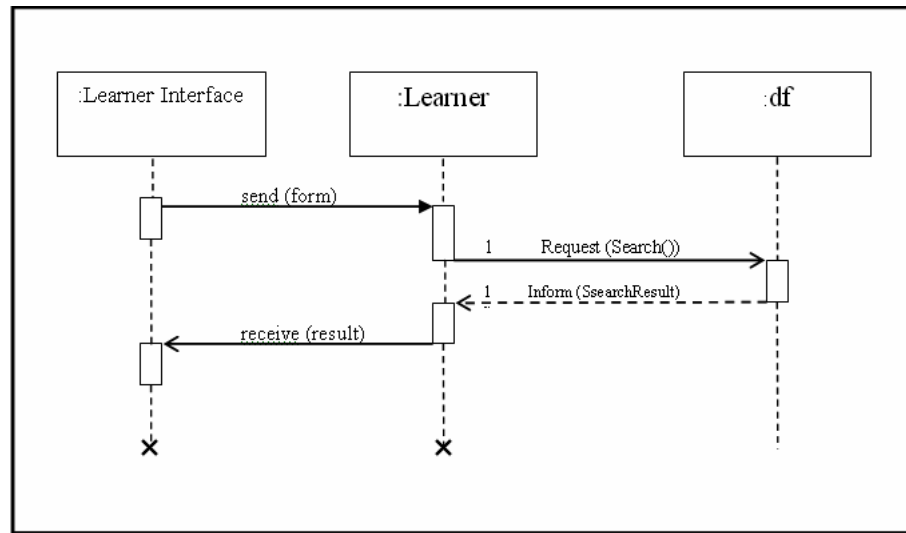


Figure 5.16: Search Request for Grid Services

## 5.7 Review of RDADeLE agents' specifications towards RDADeLE Requirements

This chapter covered the required agent types and their specifications in the RDADeLE architecture. There are five types of agents: administrative agent, regional agent, node agent, service agent and learner agent. Monitoring e-learning system is part of the management requirement which is the first requirement. The administrative agent, regional agent and node agent cooperate with each other to monitor the components of the RDADeLE architecture. The administrative agent keeps monitor the e-learning system periodically in order to manage the whole system. Monitoring is part of the first requirement which is management.

## 5.8 Summary

---

Flexibility is the second requirement of the architecture. Flexibility includes two parts: flexibility in service composition and flexibility in searching for LOs' information. Service agent is created in the e-learning system to compose services using agent to control and represent LOs' information. This is an important factor in e-learning system to become flexible in service composition. Regional agent and learner agent are used to facilitate search request from requesters.

## 5.8 Summary

This chapter has provided an overview of the architecture of the RDADeLE system. It began with account of the relationships between agents, which include the different types of communication in the RDADeLE system. Next, it described two scenarios, using the XML-based Registries Technique and the Registered-based Services Technique respectively. It then described the MAS in the RDADeLE system. This was followed by a description of the functions and descriptions of the RDADeLE agents.



# Chapter 6

## RDADeLE Implementation

### Objectives

---

- Present the RDADeLE system configuration.
  - Present validation of the simulation .
  - Present verification of the RDADeLE system.
- 

### 6.1 Introduction

There are many simulators, platforms and languages for developing grid and agent systems. These tools are usually built according to developers' aims and objectives or to users' specified requirements, i.e. they are domain specific and not based on our needs and requirements. A comparative study of these simulators and platforms has been presented in chapter 2, section 2.8, and in chapter 3, section 3.6. The JADE platform has been chosen to simulate an agent-based grid environment in order to prove the concept of building a controlled dynamic agent-based grid environment.

Building a real system is not easy, especially if the system is complicated and requires the integration of many technologies. The RDADeLE system requires the functionalities and cooperation of many parties. It also incorporates the grid and agent technologies which we name an agent-based grid. It is difficult to test the system using real parties and components, because it requires resources like regional grids and their components whose, preparing would consume time, effort and budget. Thus we have built the system using JADE which is an ideal platform to implement

## 6.2 RDADeLE System Configuration

---

our model in order to present the concepts and objectives of our research. This tool is written in Java and was built on our computational and architecture model, as described in chapter 4.

This chapter describes the implementation of the RDADeLE system [10]. It discusses the simulation of an agent-based grid environment to show our resource management and the performance of the system with and without service agents. Features of our simulation and an explanation of how the simulation works and is configured are presented in section 6.2. The system verification is demonstrated by considering the difference in performance with and without the effect of service agents, in section 6.3.

## 6.2 RDADeLE System Configuration

Before we start describing our simulation we must establish what simulation means: it is defined as “attempting to predict aspects of the behaviour of some system by creating an approximate model for it”<sup>1</sup>. There are advantages in building simulators, which include:

- There is no need to build a real system, which is not easy and requires money, time, hardware and people.
- A large number of experiments can be run and more easily controlled.
- There are possibilities for teaching, training and modelling.

The RDADeLE system prototype simulates two techniques for composing and searching for LOs’ information and grid services, as stated in chapter 5. A comparative study is presented later in this chapter. The first technique is **the XML-based**

---

<sup>1</sup>ProModel Corporation. What is simulation? <http://www.promodel.com/challenge/simulation.asp>.

## 6.2 RDADeLE System Configuration

---

**Registries Technique (XRT)**, in which LOs' information is built using XML meta-data registries. The second technique is **the Registered-based Services Technique (RST)**, whereby grid services are built using agents that are registered with the DF to be discovered. There are a number of components and mechanisms which should be included in the prototype to achieve a realistic simulated environment. The components are the main interface used to create the simulation, regional grids, grid services, information service, nodes and learners. The mechanisms are communication, termination, timing and failure.

The grid components, which are managed and controlled by agents, include regional grids, nodes and learners. Grid services are represented as agents, as we will describe next. The types of agents used in the RDADeLE system are:

- **Administrative agents:** Administrative agents create regional agents to build the RDADeLE system. They supply authentication services for regional grids and they control and manage the grid environment.
- **Regional agents:** Regional agents create regional grid components (i.e. node, learner and service agents). Each agent controls a single regional grid which includes a registry of LOs' information. The regional grid is authenticated to be registered via the administrative agent.
- **Node agents:** Each node agent controls a node in a single regional grid which represents a member staff of an institution authorised to update registries and/or node properties.
- **Service agents:** Each service agent represents a grid service in each regional grid to be discovered by learners. These agents exist only when **the RST** is used.

## 6.2 RDADeLE System Configuration

- Learner agents: A learner agent works as a facilitator for a learner who seeks LOs' information or grid services or who sends a query about the regional grid to which they are connected.

Figure 6.1 shows types of agents in the RDADeLE system.

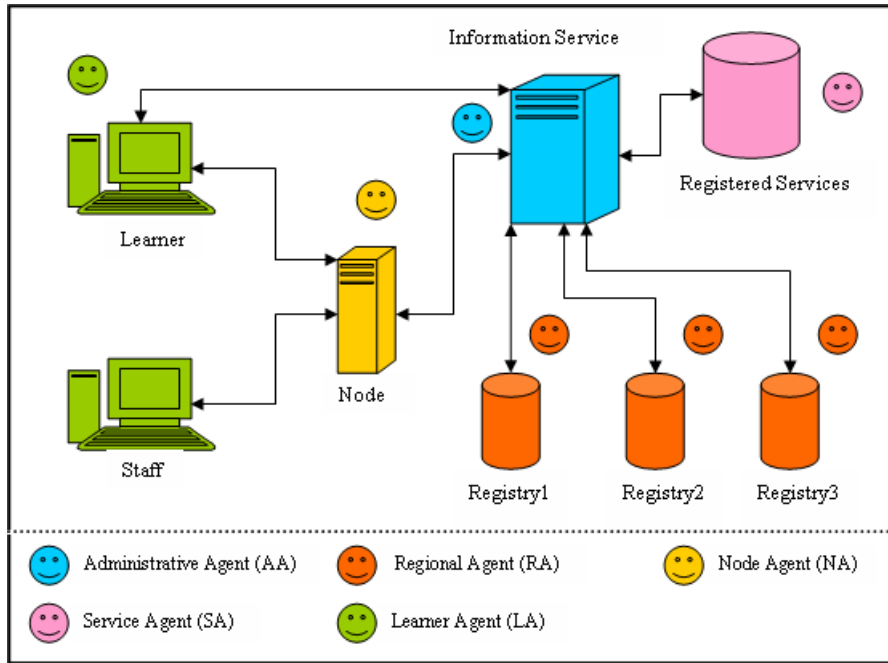


Figure 6.1: Agent Types

### 6.2.1 JADE Platform Configuration

Figure .3.2 in chapter 3, section 3.6 shows that the JADE architecture consists of the following main elements: AgentPlatform, MainContainer and Container. In our RDADeLE architecture, the main container has the following three agents by default: the Agent Management system (AMS), the Directory Facilitator (DF) and the Remote Monitoring Agent (RMA), which allows control of the life cycle of the agent platform and of all the registered agents. Besides the AMS, the DF and the RMA in the main container, we have created the administrative agent which is the first

## 6.2 RDADeLE System Configuration

type of agents to have been created for the RDADeLE system. The distributed architecture of JADE also allows remote controlling, where the GUI is used to control the execution of agents and their life cycles from a remote host. A single normal container (non-main container) represents only one regional grid in the RDADeLE system. Figure 6.2 shows the JADE Remote Agent Management GUI screen, which explains the initial status of the JADE platform, containing only the main container. These three agents control the RDADeLE system and work as an information service.

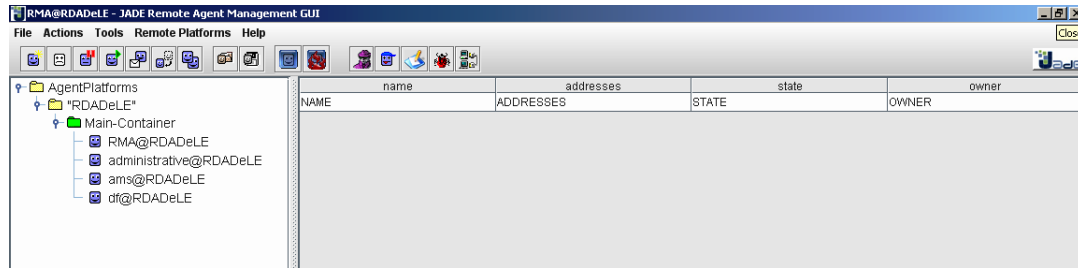


Figure 6.2: JADE Platform

### 6.2.2 Configuration of Learning Objects' information and Grid Services

The simulation was conducted using two techniques (**XRT** and **RST**) for composing and searching for LOs' information and grid services. The following subsections offer a detailed account of these techniques and the two resulting simulations:

#### 6.2.2.1 The XML-based Registries Technique (XRT)

In **XRT**, LOs' information is built as XML meta-data registries, which we assume to have been already built before running the prototype. Using this technique, there are many steps in the simulation of the agent-base grid environment. The first step is building the RDADeLE system through the GUI main interface, as shown in figure 6.3. Once the main interface has been initiated, the JADE platform launches, as

## 6.2 RDADeLE System Configuration

shown in figure 6.2. The main interface consists of the name of the system in the title bar and all fields of the system needed for the simulation, namely “Number of Regional Grids”, “# of Nodes”, “Max # Number of Learners connected to each node”, “Regional Grid Name” and the “Create” button.

Regional Grid	# of Nodes	Max # of Learners connected to each node	Regional Grid Name
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	
8	0	0	
9	0	0	
10	0	0	
11	0	0	
12	0	0	
13	0	0	
14	0	0	
15	0	0	

Create

**Figure 6.3:** The XML-based Registries Technique (XRT)

Once the number of regional grids is entered in the field marked “Number of Regional Grids”, a specification table appears on the main interface and is completed by entering the desired grid specification. The table consists of four columns to be completed in order to create the RDADeLE system. The first column is the sequence number of the regional grid “Regional Grid”, the second is the number of nodes “# of Nodes”, the third is the maximum number of learners connected to each node “Max # of Learners connected to each node” and the last is the name of the regional grid “Regional Grid Name”. Finally, the “Create” button is pressed to build the system.

Once the “Create” button has been pressed the RDADELE system and other components will be initiated and start functioning. There will be one interface for regional grid, an interface for each node agent and an interface for each learner, resulting in three types of interface, named regional interface, node interface and learner

## 6.2 RDADeLE System Configuration

interface respectively. The regional interface represents the system administrator who are authorised to add new nodes. The node interface represents the staff in an institution who are authorised to add new learners and update both the registry of the regional grid it belongs to and the node properties. The learner interface allows a learner in a particular regional grid to request LOs' information or inquire about connected regional grids at any time during the simulation.

The structure of the meta-data XML registry file of LOs' information is based on the Dublin Core Metadata Element Set (DCMES), which is the most popular metadata standard, allowing cataloguing, searching and reuse of resources. Figure 6.4 shows an example of DCMES meta-data.

```
<?xml version="1.0" encoding="UTF-8"?>
<dublincore name="Riyadh">
  <title name="Networking">
    <creator>Saber Ali</creator>
    <subject>Computing</subject>
    <description>This course is about computer networking</description>
    <date>2008-09-25</date>
    <type>example</type>
    <format>pdf</format>
    <identifier scheme="URI">http://www.computernetworking/courses/networking.pdf</identifier>
    <rights>Copyright free for non-commercial use. For commercial use, contact the creator.</rights>
  </title>
  <title name="Condensation">
    <creator>Charles R. Ward</creator>
    <subject>Chemistry</subject>
    <description>This image illustrates condensation of water vapor to liquid on the outside of a glass of water </description>
    <date>2001-09-25</date>
    <type>example</type>
    <format>image/jpeg</format>
    <identifier>ilumina:611</identifier>
    <identifier scheme="URI">http://aa.uncw.edu/digilib/chemistry/general/condensation.jpg</identifier>
    <rights>Copyright free for non-commercial use. For commercial use, contact the creator.</rights>
  </title>
</dublincore>
```

Figure 6.4: DCMS Meta-data Example

### 6.2.2.2 The Registered-based Services Technique (RST)

When the RST is used, the grid services are composed using agents, which represent grid services in the RDADeLE system. As figure 6.5 shows, there is a column in the specification table labeled “# of Grid Services” which does not appear in the main interface of the XRT. This column determines the number of grid services in each

## 6.2 RDADeLE System Configuration

regional grid in order to build grid services. In this technique, all grid services (agents) from all the regional grids are registered with the DF of the JADE platform, making it easier and more efficient for learners to search for them. However, each grid service belongs to a regional grid (i.e. the ownership of each grid service is the regional grid to which it belongs).

Regional Grid	# of Nodes	Max # of Learners connected to each node	# of Grid Services	Regional Grid Name
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	
5	0	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	
15	0	0	0	

Number of Regional Grids: 15

Create

**Figure 6.5:** The Registered-based Services Technique (RST)

The following are some properties of a grid service which are needed in composing and registering it to the DF:

- **Name:** The name of service is composed of two parts: first, the agent name of the service owner, which is the regional agent name, and second the name of agent creating the service. For example, the name of the first agent which creates the grid service in the Riyadh regional grid is named “Riyadhgridservice1”.
- **Type:** The type of service which will be presented. The type of service in our case is grid. For example, the type of any service is “Grid\_Service”.
- **Ownership:** The ownership is the regional grid to which the grid service belongs. For example, the ownership of the grid service named “Riyadhgridservice1” is “Riyadh”.



## 6.2 RDADeLE System Configuration

---

- Language: the language in which a message in the agents is expressed. In our case the language is the default language which is Semantic Language (SL0).
- Ontologies: The ontology is used in an agent message to represent agents' knowledge. All agents that share the same ontology for knowledge representation have an understanding of the word in the agent communication language.
- Protocols: These are interaction protocols for the service.
- Properties: Other agents may search the DF, based on the properties of the services provided, which have unique names and types as part of the service description.

### 6.2.3 Grid Environment Configuration

The building of this simulation includes a specification table in the main interface to determine the specifications of the regional grids. The first row specifies the number of nodes in the first regional grid, the maximum number of learners connected to each node of that grid and the regional grid name, while the second row contains the equivalent specifications for the second regional grid, and so on for the remaining regional grids. In this way, all regional grids are configured, resulting in the grid environment shown in figure 6.6. The main interface allows us to change the number of regional grids by typing in a new number in the field labeled “Number of Regional Grids” and then to add the specifications for each new regional grid. We can also edit and change each regional grid specification; hence we can change the number of nodes, the maximum number of learners connected to each node and the regional grid name.

## 6.2 RDADeLE System Configuration

Regional Grid	# of Nodes	Max # of Learners connected to each node	Regional Grid Name
1	02	07	Riyadh
2	01	04	Makkah
3	03	011	Madinah
4	02	04	AlBaha

Create

**Figure 6.6:** Regional Grid Configuration

### 6.2.4 Node Configuration

Our tool can simulate the nodes by configuring their properties. It does this by determining their names, their regionals, node types, operating system, CPU, memory, and available times. This information is assigned to the node by parsing the node property XML file which contains these properties and can be read by the node. The node name is determined automatically by the RDADeLE system, as a combination of regional name and node name. For example, “Riyadhnode1” is the name of the first node in the Riyadh regional grid. The node type can be any of the following: personal computer, mainframe, or cluster. The value of the tag “cpu” is the speed of the CPU, “memory” is the size of memory and “available\_times” is one or many time intervals during which the node will be available and can receive jobs. Figure 6.7 shows the node properties of node1 in the Riyadh regional grid.

```
<?xml version="1.0" encoding="UTF-8"?>
<node name="Riyadhnode1">
  <property name="Riyadhnode1property">
    <node_type>pc</node_type>
    <operating_system>Linux</operating_system>
    <cpu>4 cpu 3.00GHz</cpu>
    <memory>0.99GB</memory>
    <available_times>s=10:00-e=10:30,s=12:00-e=13:00</available_times>
  </property>
</node>
```

**Figure 6.7:** Node Properties

## 6.2 RDADeLE System Configuration

---

### 6.2.5 Information Service

An information service is an indispensable component in grid computing [83]. In our simulation, the main container works as an information service for the RDADeLE system. It has three default agents to control agents composing the RDADeLE system in the JADE environment. These default agents are the AMS, the DF and the RMA. Besides AMS and DF in the main container, we have created an “Administrative agent” according to the requirements of the RDADeLE system. This keeps tracks of the regional agents which control regional grids. The administrative agent helps the process of search for LO’s information by discovering all regional grids connected to the RDADeLE system.

### 6.2.6 Regional Policy Configuration

As stated in chapter 4, section 4.2, there is a policy (constraints) for each regional grid, based on regional demographic, economic, social and cultural factors. We assume that the policy of each regional grid has already been built and saved in the regional grid policy XML file. There is one policy file for each regional grid. Once a regional grid has been created and joined the RDADeLE system, the policy file is parsed to be read by the regional agent in order to be applicable. The policy file consists of many elements. These elements are demographics, society, culture, data grid policy (read, write, move, copy), application policy (use, move, copy), and grid policy (exclusive execution, add data, add application). Figure 6.8 shows the regional grid policy for the Riyadh regional grid.

## 6.3 Simulation Validation

---

```
<?xml version="1.0"?>
<regional name="Riyadh">
  <policy name="Riyadhpolicy">
    <regional_demographics>5millions</regional_demographics>
    <regional_society>public</regional_society>
    <regional_culture>eastern</regional_culture>
    <regional_data_grid_policy>read</regional_data_grid_policy>
    <regional_grid_application_policy>use</regional_grid_application_policy>
    <regional_grid_policy>add data</regional_grid_policy>
  </policy>
</regional>
```

Figure 6.8: Regional Policy

## 6.3 Simulation Validation

The simulator is validated by showing that the RDADeLE system works and does what learners really require in line with the system's aims and objectives. The following four subsections describe the methods used to validate aspects of RDADeLE and evaluate the simulation.

### 6.3.1 Verification of Grid Configuration

To verify the grid configuration we consider the following scenario. The grid is composed of three regional grids, each represented as a normal container. The first regional grid represents the Arab League and consists of at least 22 nodes (22 countries); the second represents the European Union and consists of at least 27 nodes (27 countries); and the third represents the Indian Subcontinent and consists of at least 7 nodes (7 countries).

The naming of regional grids and nodes is based on the following procedure: The Regional Grids Set (RGS) contains sequence numbers of all regional grids in the RDADeLE system and Nodes Set (NS) contains the sequence numbers of all nodes in each regional grid. An element in RGS is denoted  $(r)$ , and  $(n)$  is an element in NS. The names of the remaining nodes are based on the following equation:

### 6.3 Simulation Validation

---

$$\forall r \in \mathbf{RGS}, n \in \mathbf{NS} : regional_r node_n \quad (6.1)$$

For the purpose of verification, name of regional grid (i.e. container name) will be generated automatically based on the above equation, thus the system administrator will enter the regional grid name (i.e. the main interface does not contain the column labeled “Regional Grid Name”). Table 6.1 illustrates the grid specification and shows the corresponding agent and container names.

**Table 6.1:** Components of Regional Grids

Regional Grids and Nodes	Corresponding agent name	Container Name
Arab League	regional1	Regional-Grid1
European Union	regional2	Regional-Grid2
Indian Subcontinent	regional3	Regional-Grid3
First node (country) in Arab League	regional1node1	Regional-Grid1
Second node (country) in Arab League	regional1node2	Regional-Grid1
First node (country) in European Union	regional2node1	Regional-Grid2
Second node (country) in European Union	regional2node2	Regional-Grid2
First node (country) in Indian Subcontinent	regional3node1	Regional-Grid3
Second node (country) in Indian Subcontinent	regional3node2	Regional-Grid3

Figure 6.9 shows the main interface, which contains the specification of a grid completed in order to create the prototype of the scenario.

Figure 6.10 shows the main container and non-main containers which correspond to the regional grids of the scenario.

Figure 6.11 shows some of the components of the first non-main container, which correspond to the nodes and learners of the Arab League regional grid.

Figure 6.12 shows some of the components of the second non-main container,

### 6.3 Simulation Validation

RDADeLE System

Number of Regional Grids  
3

Regional Grid	# of Nodes	Max # of Learners connected to each node
1	022	04
2	027	05
3	07	03

Submit

Figure 6.9: Regional Grid Verification

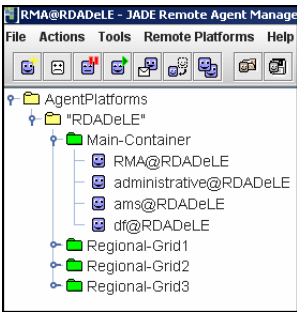


Figure 6.10: All Regional Grids (Containers)

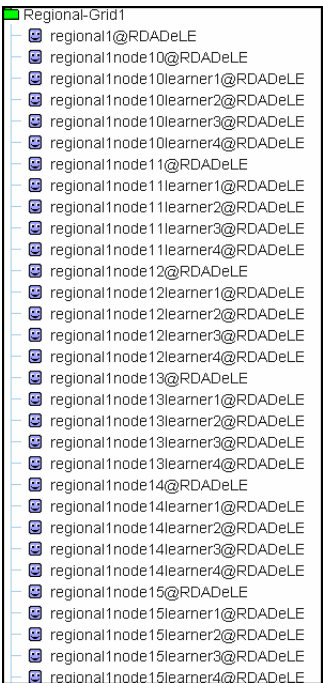
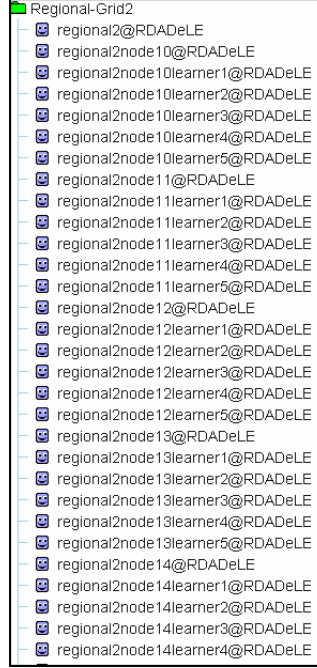


Figure 6.11: Components of Arab League

### 6.3 Simulation Validation

---

which correspond to the nodes and learners of the European Union regional grid.



**Figure 6.12:** Components of European Union

Figure 6.13 shows some of the components of the third non-main container, which correspond to the nodes and learners of the Indian Subcontinent regional grid.

#### 6.3.2 Verification of LOs' Information and Grid Service Configuration

As stated earlier, there are two techniques in building and accessing LOs' information and grid services in the RDADeLE system: **XRT** and **RST**. In **the XRT**, LOs are reached by searching their metadata registry XML files. Registries are read by both the regional agent and the node agent if needed. On one hand, the regional agent reads its registry file to be ready for learners to search for LOs' information, while the node agent reads its regional grid registry for it to be maintained (read, write, update,...) in order to bring the registries up to date for searching. Only authorised members of staff are eligible and responsible for maintaining the registries. Figure

### 6.3 Simulation Validation

---



**Figure 6.13:** Components of Indian Subcontinent

6.4 shows the DCMES meta-data example adopted here.

Figure 6.14 illustrates a node interface, which shows content of registry in the “regional1” regional grid on the screen after reading them in response to the command *show*.

In **The RST**, grid services are reached by searching for registered service agents in the DF. In this technique, grid services are built as agents. For the purpose of research we have already built these limited services as follows:

1. Create an agent named “GridService” as a grid service.
2. Determine service description, which includes (*Type*) and (*Name*). The (*Type*) of service we intend to build to be embedded in the agent is “Grid\_Service”. The *Name* of the service is a combination of localname and type.
3. Determine properties (keywords) of the service. The properties consist of two elements, which are (*Name*) and (*Value*). In our case the (*Name*) is “Conden-



## 6.3 Simulation Validation

---

```
regional1node1>show
[
Regional: regional1 :
name: Networking
creator: Saber Ali
subject: Computing
description: This course is about computer networking
date: 2008-09-25
type: example
format: pdf
identifier scheme = URI: http://www.computernetworking/courses/networking.pdf
rights: Copyright free for non-commercial use. For commercial use, contact the creator.
name: Condensation
creator: Charles R. Ward
subject: Chemistry
description: This image illustrates condensation of water vapor to liquid on the outside of a glass of water
date: 2001-09-25
type: example
format: image/jpeg
identifier: ilumina:611
identifier scheme = URI: http://aa.uncw.edu/digilib/chemistry/general/condensation.jpg
rights: Copyright free for non-commercial use. For commercial use, contact the creator.

regional1node1>
```

Figure 6.14: Registry Content List

sation” and the (Value) is “Chemistry”.

4. Add the above service to the “GridService” agent using the *addServices()* method.
5. Register the agent with the DF using the *register* method.

Figure 6.15 shows the Java code which builds and registers the grid service.

### 6.3.3 Verification of Searching for Learning Objects’ Information and Grid Services

Searching for LOs’ information and grid services is performed using a similar command pattern in both techniques, as follows *title:subject* through the learner interface. The keyword *title* is the lesson name, and the *subject* is the module or the course name.

The **XRT** technique provides a flexible way to search and restore LOs’ information from all over the grid environment using regional grid registries. This technique searches for corresponding words in tags, titles and subjects using the keywords

### 6.3 Simulation Validation

---

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd = new ServiceDescription();
sd.setType("Grid_Service");
sd.setName(getLocalName()+sd.getType());
sd.setOwnership(ownerName);

Property p1 = new Property();
p1.setName("Condensation");
p1.setValue("Chemistry");
sd.addProperties(p1);

dfd.addServices(sd);

try {
    DFService.register( this, dfd );
}
catch (FIPAException fe) {
    fe.printStackTrace();
}
```

**Figure 6.15:** Java Code for Building and Registering Grid Service

above. The search looks for the *title* keyword first, then for the *subject* if this is not found. The result is composed of two parts:

1. The name of the regional grid where the keywords are found.
2. The keywords themselves, *title* and *subject* which are searched for by a learner.

For the purpose of verification we conducted the test twice, for two scenarios. In the first scenario, the RDADeLE system had only one regional grid (one registry file). The content of the registry of this regional grid is shown in figure 6.4, while figure 6.16 shows the search result using the learner interface.

The second scenario is that the RDADeLE system has 3 regional grids (3 registry files). The content of the registry of this regional grid is shown in figure 6.4, except that the value of the “subject” tag under the tile “Networking” is “Computing3” instead of “Computing”. Figure 6.17 shows the search result using the learner interface.

The **RST** technique provides a fast way of searching for pre-registered services (grids in our case) which may lead to LO materials. These grid services are registered

## 6.3 Simulation Validation

---

```
regional1node1learner2>
regional1node1learner2>Networking:Security
regional1node1learner2>
The result from: regional1 :

title: Networking
subject: Computing

regional1node1learner1>Security:Computing
regional1node1learner1>
The result from: regional1 :

title: Networking
subject: Computing

regional1node1learner1>Networking:Computing
regional1node1learner1>
The result from: regional1 :

title: Networking
subject: Computing

regional1node1learner1>Security:Security
regional1node1learner1>
regional1node1learner2>
No matched result from regional : regional1

regional1node1learner2>
regional1node1learner2>
```

**Figure 6.16:** Verification of XML Search in One Registry

```
regional3node2learner2>Networking:Computing
regional3node2learner3>
The result from: regional1 :

title: Networking
subject: Computing

regional3node2learner2>Based on the regional2 policies, it is not permited to access the data grid.
The result from: regional3 :

title: Networking
subject: Computing3

regional3node2learner3>
```

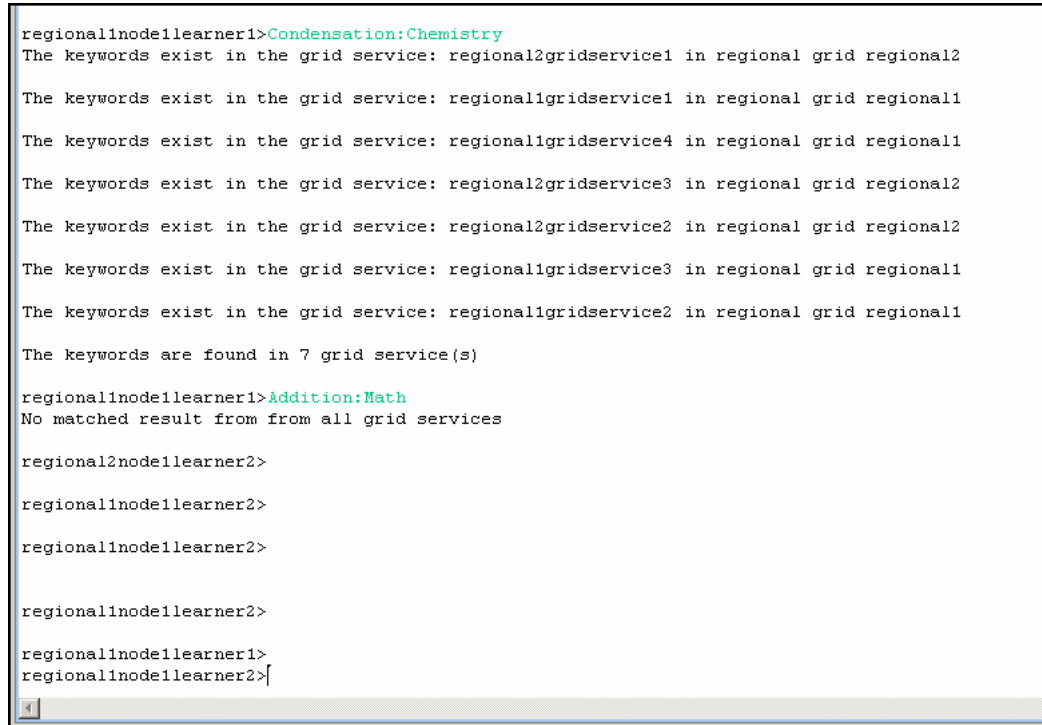
**Figure 6.17:** Verification of XML Search in Three Registries

### 6.3 Simulation Validation

---

with the DF. For the purpose of the verification test, all registered grid services are the same. The properties of grid services comprises *title* and *subject*, which have the values “Condensation” and “Chemistry” respectively (See figure 6.15). We assume that the RDADeLE system is composed of only two regional grids and that the number of registered grid services in the first and second regional grids is 4 and 3 respectively. The total number of registered grid services is thus 7.

Figure 6.18 shows a learner’s search for registered grid services and the result through the learner interface.



```
regional1node1learner1>Condensation:Chemistry
The keywords exist in the grid service: regional2gridservice1 in regional grid regional2

The keywords exist in the grid service: regional1gridservice1 in regional grid regional1

The keywords exist in the grid service: regional1gridservice4 in regional grid regional1

The keywords exist in the grid service: regional2gridservice3 in regional grid regional2

The keywords exist in the grid service: regional2gridservice2 in regional grid regional2

The keywords exist in the grid service: regional1gridservice3 in regional grid regional1

The keywords exist in the grid service: regional1gridservice2 in regional grid regional1

The keywords are found in 7 grid service(s)

regional1node1learner1>Addition:Math
No matched result from from all grid services

regional2node1learner2>

regional1node1learner2>

regional1node1learner2>

regional1node1learner2>

regional1node1learner1>
regional1node1learner2>[
```

**Figure 6.18:** Verification of Registered Services Search

#### 6.3.4 Verification of Regional Policy Application

Regional policy has been configured in the RDADeLE system in order to provide autonomy for the regional grids. At the same time, the policy controls the data flow between the RDADeLE system components. Verification tests were conducted for

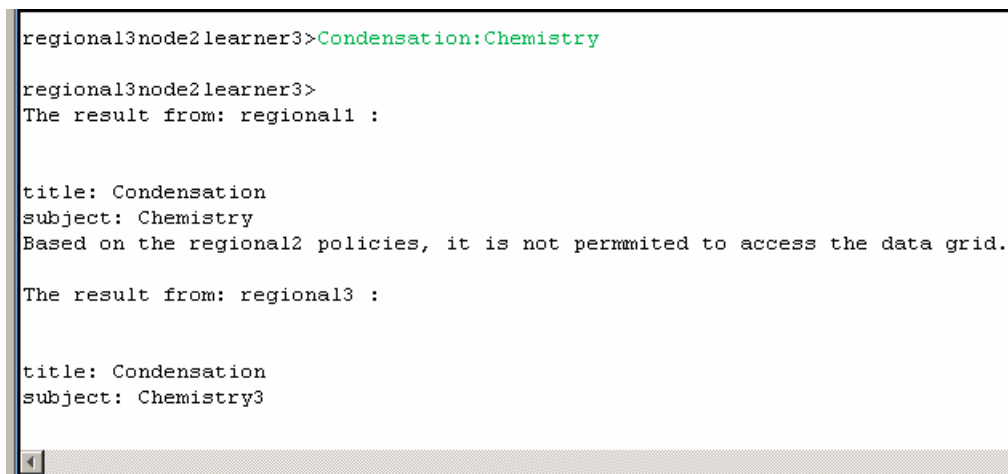
### 6.3 Simulation Validation

---

both the **XRT** and the **RST** techniques; however, we believe that more research needs to be conducted into regional policy, which we consider to be part of our future work.

The verification of regional policy application was tested under the following assumptions for both techniques: the number of regional grids created is 3 and the value of the ‘data grid policy’ element in the policy files of the first and third regional grids is *read*, a value which permits the reading of the data grid from the corresponding regional grid, whereas the value of this element in the policy file of the second regional grid is *notread*, which does not permit the data grid to be read from the corresponding regional grid.

Figure 6.19 shows the result of applying the regional policy in the **XRT technique**.



```
regional3node2learner3>Condensation:Chemistry
regional3node2learner3>
The result from: regional1 :

title: Condensation
subject: Chemistry
Based on the regional2 policies, it is not permited to access the data grid.

The result from: regional3 :

title: Condensation
subject: Chemistry3
```

**Figure 6.19:** Applying Regional Policy in the XRT Technique

Figure 6.20 shows the result of the application of the regional policy in the **RST** technique, assuming that 3 grid services belong to regional grid 1, 4 grid services to regional grid 2 and 5 to regional grid 3.

## 6.4 Review of RDADeLE implementation towards RDADeLE Requirements

---



```
regional3node3learner2>Condensation:Chemistry
The keyword exists in the grid service: regional3gridservice4

Based on the regional policies, it is not permitted to access the data grid in regional2gridservice1

The keyword exists in the grid service: regional1gridservice1

The keyword exists in the grid service: regional3gridservice5

Based on the regional policies, it is not permitted to access the data grid in regional2gridservice3

Based on the regional policies, it is not permitted to access the data grid in regional2gridservice2

The keyword exists in the grid service: regional3gridservice2

The keyword exists in the grid service: regional1gridservice3

The keyword exists in the grid service: regional1gridservice2

The keyword exists in the grid service: regional3gridservice1

The keyword exists in the grid service: regional3gridservice3

Based on the regional policies, it is not permitted to access the data grid in regional2gridservice4

The keywords are found in 12 grid service(s)

regional3node3learner2>
```

Figure 6.20: Applying Regional Policy in the RST Technique

## 6.4 Review of RDADeLE implementation towards RDADeLE Requirements

This chapter covered the implementation of RDADeLE prototype. The developed prototype used in the implementation has been validated and verified against some factors to build a similar e-learning environment, conduct enough experiments, and obtain results from experiments to judge on RDADeLE architecture. Grid environment has been configured to meet the first and the second requirements management and extensibility.

The third requirement is the flexibility. Verification of composition of LOs' information and service has been conducted to proof that the RDADeLE architecture is more flexible using both techniques XRT and RST. Meanwhile, verification of searching for LOs' information and services has been conducted to proof that the RDADeLE architecture is more flexible using both techniques XRT and RST.

## 6.5 Summary

This chapter has described the simulation of the RDADeLE system as an agent-based grid environment. It explains how the simulation was configured and how it worked, including configuration of LOs' information and grid services, nodes, information services and regional policy. Validation of the simulation included verification of grid configuration, of both LOs' information and grid service configuration, verification of the search process and regional policy.

# Chapter 7

## Results and Evaluation

### Objectives

---

- Present the specifications of the testing environment.
  - Present the RDADeLE system scalability, reliability and efficiency in searching for LO's information and grid services.
  - Discuss evaluation of scalability, reliability and efficiency of the RDADeLE system.
- 

### 7.1 Introduction

The study of the results of this thesis is presented in this chapter including the evaluation of the RDADeLE system in scalability, reliability and efficiency in searching for LO's information and grid services. The two techniques of composing and searching for LOs' information and grid services, **the XRT technique** and **the RST technique**, have been evaluated.

Firstly, scalability in the RDADeLE system has been examined against some factors such as number of regional grids and number of registered services agents. Those factors has been evaluated against memory consumption. Secondly, reliability of the RDADeLE system has been tested which includes fault-tolerance using replicated information service, type of connections between regional grids and how these connections are maintained in case of a failure. Finally, efficiency in searching for LOs' information and grid services has been evaluated. The efficiency has been tested to



## 7.2 Deployment Environment Setup

---

evaluate an increase in the number of registries and grid service agents against mean search time in milliseconds.

We have adopted the RDADeLE system in Saudi Arabia as a case study in our evaluation. This provides us a clear view of the impact of the RDADeLE system in controlling and managing e-learning systems within the Saudi Arabian environment in particular, and all over the world in general.

## 7.2 Deployment Environment Setup

We conducted experiments in our simulation with different regional grids, grid services and LOs' information. All of these components have been modeled and controlled using agent technology. Each regional grid has many nodes which are connected to it and each node has many learners which are connected to it. The goal of our simulation is to analyse the impact of agent technology within the grid environment on e-learning systems in the Kingdom of Saudi Arabia. For the evaluation we used a desktop PC with the following specifications:

Pentium (R) 4 CPU IV, 3.0 GHz, 0.99 GB RAM

Microsoft Windows XP Professional (SP2)

Java 1.6.0\_11

JADE v3.5

## 7.3 Scalability

In telecommunications and software engineering, scalability is a desirable property to have in a system, a network, or a process, which indicates its ability to either handle growing amounts of work in an efficient manner, or to be readily enlarged. For

### 7.3 Scalability

---

example, it can refer to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added. The scalability is an important feature which exposes limitations of a system [86].

The scalability of our RDADeLE system has been examined against factors such as number of regional grids and number of grid services.

#### 7.3.1 Number of Regional Grids (Non-Main Containers)

In our case study there are 13 administrative regions in the Kingdom of Saudi Arabia. The RDADeLE system in this case has 13 regional grids. Each regional grid is represented as a non-main container. Regional agents play a major role in the overall performance of the RDADeLE system. One of the regional agents' roles is the provision of search process of LOs' information in **the XRT technique**. The number of regional agents in our case study could be more than 13, which depends upon the number of learners within each regional grid. If there is an increase in the number of learners within a single regional grid the RDADeLE system increases the number of regional agents in order to serve learners' requests. As a result of the increase in the number of regional agents the RDADeLE system will extend the regional grid by dividing it into sub-regional grids (sub-system). We assume that one regional agent serves no more than 3000 elements (i.e. node, learner and grid service agents).

For the purpose of testing we reduce the maximum number of elements which are served by one regional agent to 100. The reason for this reduction is that we do not have the sufficient resources to accomplish the experiment and 100 elements is deemed to be sufficient to prove the concept through our case study. We assume that we are building the system which composes of two regional grids, "Riyadh" and "Makkah". We will create more than 100 elements (260 in our case) in the regional grid "Riyadh" to show the action of the RDADeLE system as shown in figure 7.1.

### 7.3 Scalability

Whereas the “Makkah” regional grid will accommodate less than 100 learners (6 in our case).

The screenshot shows the 'RDADeLE System' window. At the top, there is a label 'Number of Regional Grids' with a text box containing the value '2'. Below this is a table with the following data:

Regional Grid	# of Nodes	Max # of Learners connected t...	# of Grid Services	Regional Grid Name
1	10	020	050	Riyadh
2	01	02	03	Makkah

At the bottom of the window is a 'Create' button.

**Figure 7.1:** The Case Study Data Using RST Technique

Once we press the “Create” button, the RDADeLE system will extend the “Riyadh” regional grid to become three sub-regional grids “Riyadh0”, “Riyadh1” and “Riyadh2” as shown in figure 7.2.



**Figure 7.2:** Extension of Riyadh Regional Grid

Eventually each sub-regional grid, “Riyadh0”, “Riyadh1” and “Riyadh2”, accommodates a certain number of nodes, learners and grid services. In relation to the number of nodes, learners and grid services the “Riyadh0” and “Riyadh1” sub-regional grids accommodate 80 learners, 4 nodes and 17 grid services each. Whereas regional grid “Riyadh2” accommodates 40 learners, 2 nodes and 16 grid services.

The following algorithm shows the mechanism of scalability of the RDADeLE system during the creation of the system considering the *NumberOfNodes*, *NumberOfLearners* and *NumberOfGridServices* entered by the system administrator:

### 7.3 Scalability

---

#### Algorithm 6 Scalability Algorithm

---

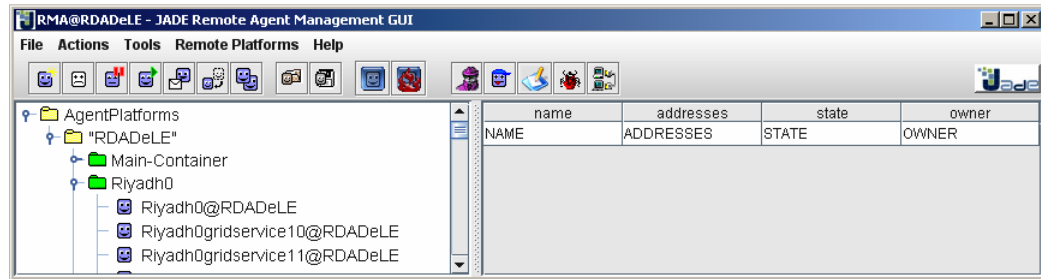
```

1: for each regional grid do
2:    $totalNumberofagents \leftarrow (NumberOfNodes * NumberOfLearners) +$ 
      $NumberOfNodes + NumberOfGridServices$ 
3:   if  $totalNumberofagents > maxNumberOfAgents$  then
4:      $numOfSubRegional \leftarrow ceiling(totalNumberofagents / maxNumberOfAgents)$ 
5:      $interval1 \leftarrow ceiling(NumberOfNodes / numOfSubRegional)$  {determine
       number of nodes for each sub-regional grid}
6:      $interval3 \leftarrow ceiling(NumberOfGridServices / numOfSubRegional)$ 
       {determine number of grid services for each sub-regional grid}
7:     for each sub-regional grid do
8:       determine sub-regional grid name (RegionalNameNew)
9:       Call Administrative.createRegional (interval3, interval1,
       NumberOfLearners, RegionalNameNew)
10:    end for
11:   else
12:     Call Administrative.createRegional (NumberOfGridServices,
       NumberOfNodes, NumberOfLearners, RegionalName)
13:   end if
14: end for

```

---

The sub-regional grid “Riyadh0” is served by the regional agent “Riyadh0@RDADeLE” as shown in figure 7.3.



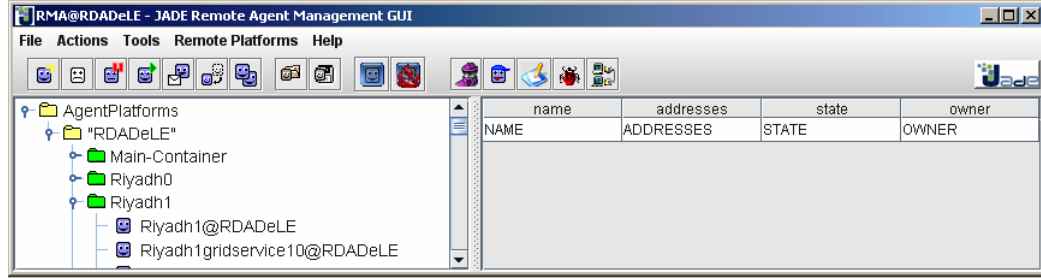
**Figure 7.3:** Regional agent “Riyadh0@RDADeLE” Serving Sub-regional Grid “Riyadh0”

The sub-regional grid “Riyadh1” is served by the regional agent “Riyadh1@RDADeLE” as shown in figure 7.4.

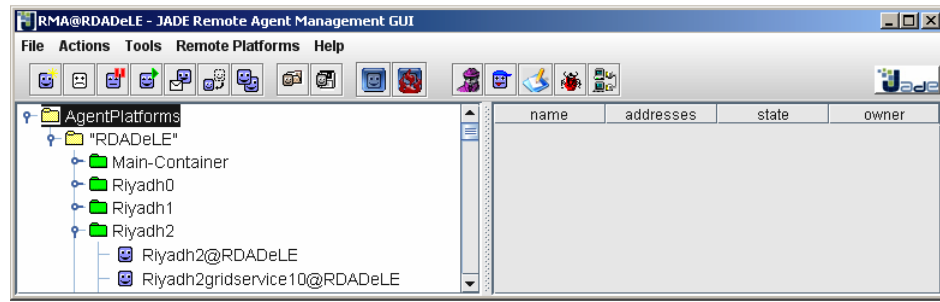
Finally, the sub-regional grid “Riyadh2” is served by the regional agent “Riyadh2@RDADeLE” as shown in figure 7.5.

For the purpose of testing the RDADeLE system, Table 7.1 illustrates the scalability of a number of regional grids the RDADeLE system can generate using **the**

### 7.3 Scalability



**Figure 7.4:** Regional agent “Riyadh1@RDADeLE” Serving Sub-regional Grid “Riyadh1”



**Figure 7.5:** Regional agent “Riyadh2@RDADeLE” Serving Sub-regional Grid “Riyadh2”

**XRT technique**, assuming that each regional grid (non-main container) has only one node and one learner. Moreover, table 7.1 shows the volume of memory consumption according to the number of regional agents.

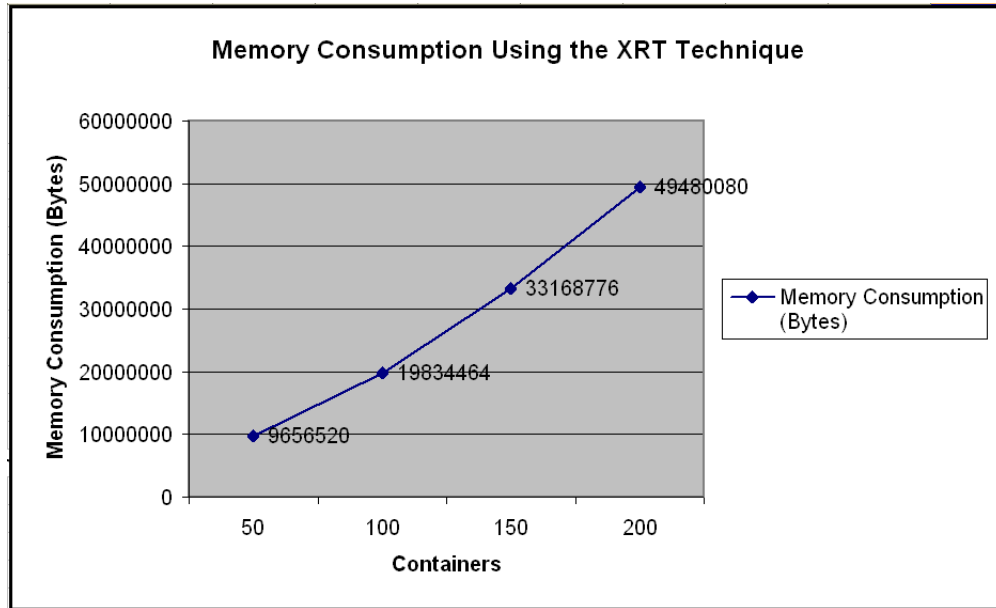
**Table 7.1:** Scalability of Number of Regional Grids Using the XRT Technique

Containers	Memory Consumption (Bytes)
50	9656520
100	19834464
150	33168776
200	49480080

The graph in figure 7.6 shows the relationship between increasing the volume of memory consumption against the number of regional agents using **the XRT technique**.

The error message in figure. 7.7 appears at regional grid number 238. This message indicates that there are insufficient resources used in the experiment (i.e. no

### 7.3 Scalability



**Figure 7.6:** Number of Regional Grids Against Memory Consumption Using the XRT Technique

available space in JVM which is limited by the available memory).

```
regional238 Successfully registered.
07-Jun-2009 15:02:57 jade.core.messaging.MessagingService clearCachedSlice
INFO: Clearing cache
07-Jun-2009 15:03:00 jade.core.PlatformManagerImpl$1 nodeAdded
INFO: --- Node <Regional-Grid243> ALIVE ---

regional237node1>07-Jun-2009 15:03:13 jade.core.messaging.MessageManager$Deliverer run
WARNING: MessageManager cannot deliver message (INFORM sender: ( agent-identifier :name administrative@RDADeLE
java.lang.OutOfMemoryError: Java heap space
*** Uncaught Exception for agent regional203node1learner1 ***ERROR: Agent regional203node1learner1 died witho
State was 2
java.lang.OutOfMemoryError: Java heap space
```

**Figure 7.7:** Error of Maximum Number of Regional Grids Using the XRT Technique

Table 7.2 illustrates the scalability of a number of regional grids the RDADeLE system can generate using **the RST technique** assuming that each regional grid (non-main container) has only one node, one learner, and one service.

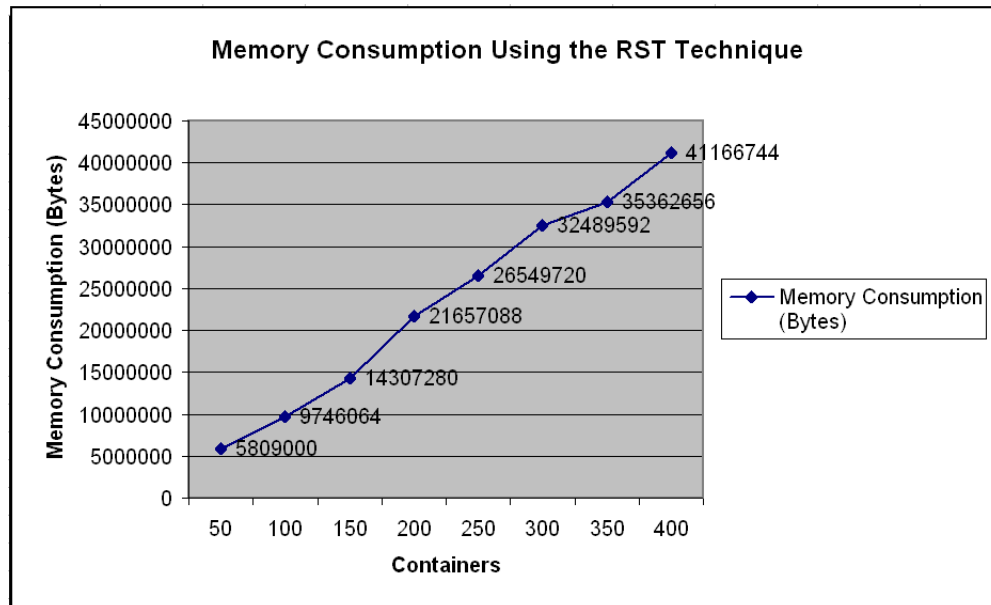
The graph in figure 7.8 shows the relationship between increasing memory consumption and the number of regional grids using **the RST technique**.

The error message in figure 7.9 appears at regional grid 415. This message indicates that there are insufficient resources used in the experiment (i.e. no available space in JVM which is limited by the available memory).

### 7.3 Scalability

**Table 7.2:** Scalability of Number of Regional Grids Using the RST Technique

Containers	Memory Consumption (Bytes)
50	5809000
100	9746064
150	14307280
200	21657088
250	26549720
300	32489592
350	35362656
400	41166744



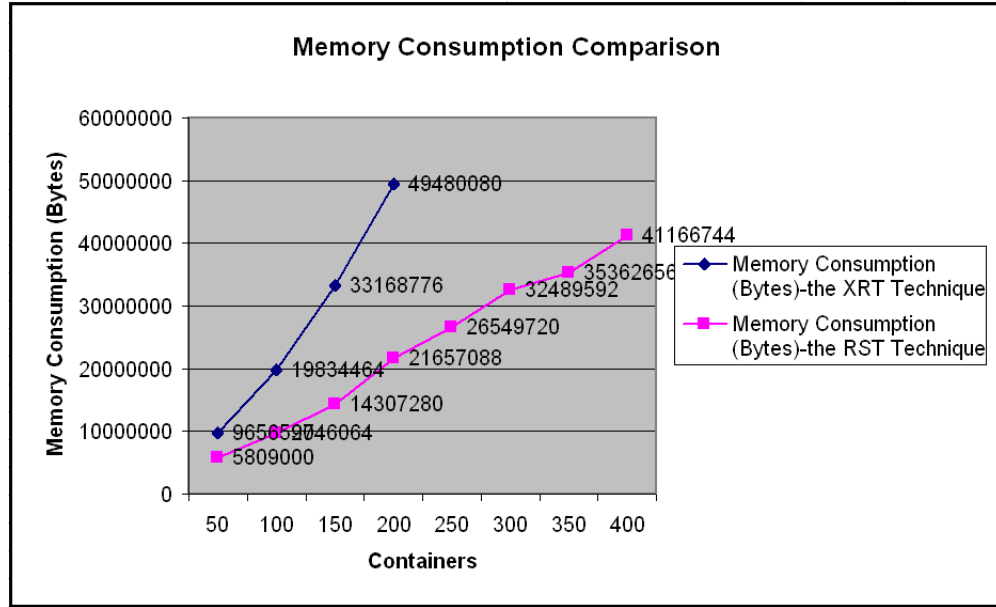
**Figure 7.8:** Number of Regional Grids Against Memory Consumption Using the RST Technique

```
regional415node1learner1>
regional415node1learner1>java.lang.OutOfMemoryError: Java heap space
at java.lang.Class.getDeclaredMethods0(Native Method)
```

**Figure 7.9:** Error of Maximum Number of Regional Grids Using the RST Technique

### 7.3 Scalability

The graph in figure 7.10 shows the difference in memory consumption using both techniques, **the XRT** and **the RST**. It shows that more regional grids can connect to the RDADeLE system with a decrease in memory consumption using **the RST technique**.



**Figure 7.10:** Comparison of Regional Grid Scalability using the XRT and the RST Techniques

#### 7.3.2 Number of Grid Services Agents

All grid services agents with their descriptions are registered with the DF. The DF is assumed to be located in the Riyadh administrative region where the RDADeLE platform is initiated. The DF is accessed by all learners from all 13 administrative regions. For the purpose of testing we assume that the number of regional grids in the following test is 1. Table 7.3 illustrates the scalability of the number of grid services (with their descriptions) which are registered with the DF, the amounts of memory these descriptions can allocate, and number of threads.

The maximum number of grid services agents that can be created in one regional



## 7.3 Scalability

**Table 7.3:** Scalability of Number of Grid Services Agents

Grid Services	Memory Consumption (Bytes)	Number of Threads
100	486312	138
200	1526928	238
300	1987152	339
400	2299344	438
500	4163584	540
600	4164728	641
700	4901768	739
800	5673232	840
900	6470984	938
1000	7237992	1039
1500	10760088	1537
2000	14135256	2034

grid (non-main container) is 2365 services. The error in figure 7.11 occurs when more than 2365 services agents are created. This message indicates that there are insufficient resources used in the experiment (i.e. no available space in JVM which is limited by the available memory).

```
The grid service: ( regionalgridservice2363Grid_Service ) has been added to Directory Facilitator (DF).  
The grid service: ( regionalgridservice2364Grid_Service ) has been added to Directory Facilitator (DF).  
The grid service: ( regionalgridservice2367Grid_Service ) has been added to Directory Facilitator (DF).  
The grid service: ( regionalgridservice2365Grid_Service ) has been added to Directory Facilitator (DF).  
Exception in thread "RHI TCP Connection(idle)" Exception in thread "Deliverer-1" java.lang.OutOfMemoryError: Java heap space  
at java.util.Arrays.copyOf(Unknown Source)
```

**Figure 7.11:** Error of Maximum Number of Registered Grid Services in a Regional Grid

### 7.3.3 Discussion

In order to reduce the limitations of the RDADeLE system which have appeared in the previous experiments, we suggest the following:

- Since the JADE platform is a distributed platform we can activate several non-main containers on different hosts and spread agents across them in each administrative region. This could be deployed as each non-main container corresponds to each institution within each administrative region. For instance,

### 7.3 Scalability

---

there could be more than 252 non-main containers which correspond to the total number of institutions among the 13 administrative regions as shown in table 4.2. In this way we attain a highly scalable system. This is the approach adopted in critical applications in Telecom Italia.

- It is not recommended that we create a high number of agents in one container (for example 5000 agents). We should spread agents across different containers considering the above mentioned point about containers across different hosts. JADE is highly scalable on top of distributed architectures and we want to avoid placing everything into a single Java Virtual Machine (JVM). Non-main containers can then be launched on the same host, or on remote hosts, that connect themselves with the main container of the agent platform, resulting in a distributed system that appears to be a single agent platform from the outside.
- We can increase the number of services agents registered with the DF by keeping the DF catalogue in a database, for instance the DF can be configured to store its catalogue directly into a database (e.g. Oracle database management system) which enables us to register millions of agents with the DF. The following command line will launch a JADE main container with a DF that will store the DF catalogue into a HSQLDB<sup>1</sup> database. The database will be started automatically together with JADE, provided that the HSQLDB libraries can be found in the CLASSPATH [45].

```
java jade.Boot -jade_domain_df_db-default
```

**Figure 7.12:** The Command Line to Store DF Catalogue into a Database

---

<sup>1</sup>HSQLDB (Hyper Structured Query Language Database) is a relational database management system written in Java.

## 7.4 Reliability

---

- We may eliminate learner agents from the RDADeLE system and allow learners to use their own available resources (e.g. PCs) to communicate with other RDADeLE agents.
- The graph in figure 7.10 shows the difference in memory consumption using both techniques, **the XRT** and **the RST**. It shows that using **the RST technique** the RDADeLE environment can be built with more regional grids with less memory consumption.

## 7.4 Reliability

The RDADeLE system should maintain reliability in order to bind all administrative regions of the Kingdom of Saudi Arabia with each other and with their components. Maintaining reliability insures that the RDADeLE system is robust enough to control all administrative regions. An introduction to system reliability in chapter 4, section 4.6 has been presented. Keeping track of administrative, regional, node, learner and service agents maintains information service within the main container which is located in the Riyadh administrative region. JADE distributed architecture relies on the main container to coordinate all other components (regional grids in our case) and to keep together the whole platform. Though most JADE operations are decentralized, there are some essential features that are supported only by the main container. These features are [45]:

- Managing the Container Table which is the set of all the components that comprise the distributed platform.
- Managing the Global Agent Descriptor Table which is the set of all the agents hosted by the distributed platform, together with their current location.

## 7.4 Reliability

---

- Managing the Message Transport Protocol (MTP) table which is the set of all deployed MTP endpoints, together with their deployment location.
- Hosting the platform Agent Management System (AMS) agent and the Directory Facilitator (DF) agent.

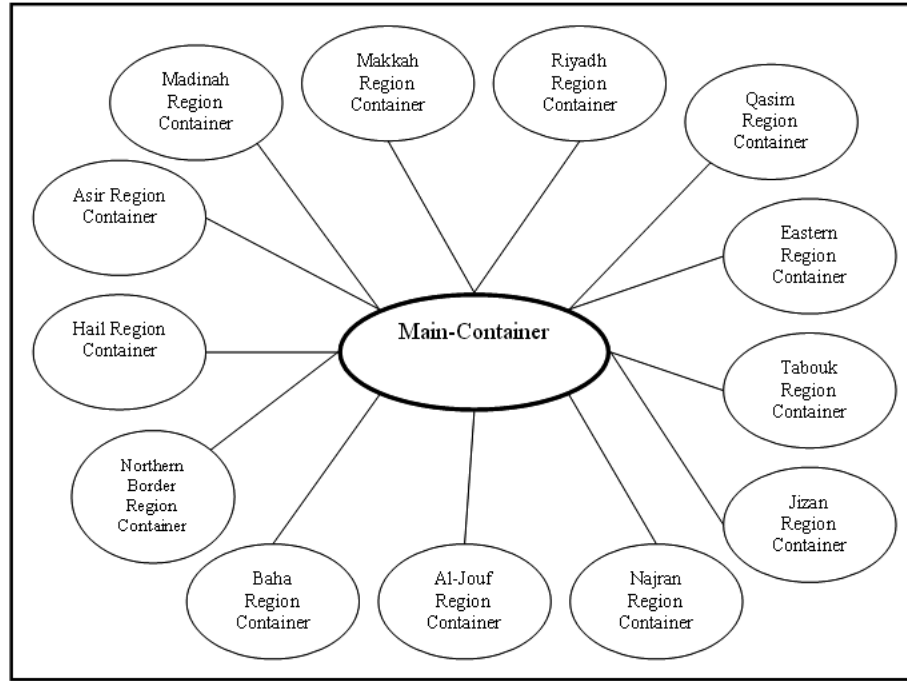
If the main container terminates or otherwise becomes unavailable to the other platform containers, all the above features will thus become unavailable and this severely hampers platform operations.

To keep the RDADeLE system fully operational, even in the event of a failure of the main container, main container replication is adopted which is part of the JADE feature. Using this technique, it is possible to start any number of main containers (a master main container actually holding the AMS and a number of backup main containers), which will arrange themselves in a logical ring so that whenever one of them fails, the others will notice and act accordingly. Non-main containers will then be able to connect to the platform through any of the active main containers; the different copies of the main container will evolve together using cross-notification.

Figure 7.13 shows that the RDADeLE system has a star topology without main container replication, while figure 7.14 shows the topology of the RDADeLE system changes into a ring of stars with main container replication.

Figure 7.14 shows that three main container nodes are arranged in a ring, and each node monitors its neighbor in the fault-tolerant configuration. Moreover, the figure shows that non-main containers can be arbitrarily spread among the available main container nodes. Any non-container is connected to exactly one main container node and in the absence of failures it is completely unaware of all the other copies. If the node Main-Container-1 fails, the node Main-Container-2 will notice and inform all the other main container nodes (in this case only the Main-Container). Then, a

## 7.4 Reliability



**Figure 7.13:** Fault Tolerance Without Replicated Main Containers

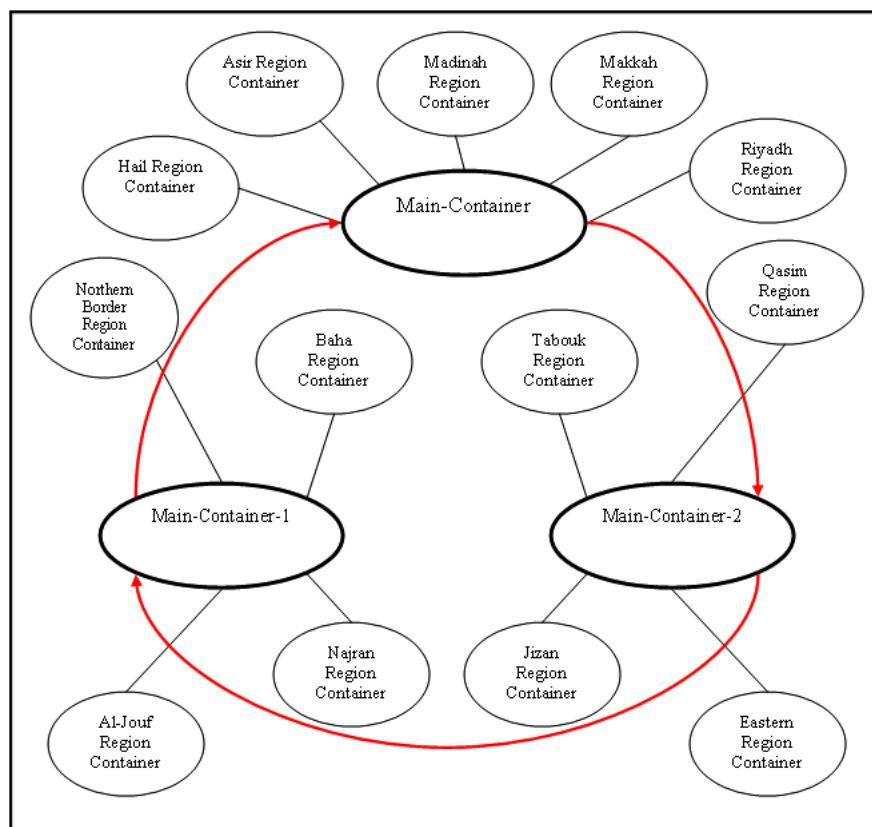
smaller ring will be rebuilt with the remaining main container nodes (Main-Container and Main-Container-2).

When a main container node fails, there will generally be some orphaned non-main containers, in our case study, supposing Main-Container-1 fails, Northern Border Region Container, Baha Region Container, Al-Jouf Region Container and Najran Region Container will become orphans. When an orphan container detects that its main container node is not available anymore, it attaches itself to another main container node, for this to succeed, a non-main container must know the list of all the main container nodes present in the platform.

JADE supports two policies in distributing the main container node list to non-main containers. The first option is to activate the Address-Notification service on all main container nodes and on the non-main containers. This service will detect additions and removals to the main container nodes' ring and update the address

## 7.4 Reliability

---



**Figure 7.14:** Fault Tolerance with Replicated Main Containers

## 7.4 Reliability

---

list of all non-main containers involved. The second option is to pass the address list to a starting non-main container with the (*-smaddrs*) command line argument. This approach avoids generating notification traffic towards non-main containers but assumes a fixed list of main container nodes, which is known beforehand.

In order to build a fault tolerant platform we start by activating replication services on the master main container which is named Main-Container in a host machine named host1 assuming that there are two copies of main container, named Main Container-1 and Main Container-2, on two other host machines named host2 and host3 respectively.

The command in figure 7.15 starts a master main container node (i.e. Main-Container) on machine host1 and activates the Main-Replication and Address-Notification services on it. The name of the platform is RDADeLE which is determined by the option (*-name*).

```
host1: java jade.Boot -name RDADeLE -services jade.core.replication.MainReplicationService;  
jade.core.replication.AddressNotificationService
```

**Figure 7.15:** Activating Replication Services on Master Main Container

The command in figure 7.16 uses the (*backupmain*) option to specify that the newly started node (i.e. Main-Container-1) is a main container, but does not make a new platform on its own. Rather, this new node is to join an existing platform that is specified by the (*-host*), (*-port*) and (*name*) options.

```
host2: java jade.Boot -backupmain -local-port 1234 -host host1 -port 1099 -name RDADeLE  
-services jade.core.replication.MainReplicationService;  
jade.core.replication.AddressNotificationService
```

**Figure 7.16:** Generating Replication of Main-Container Producing Main-Container-1

The command in figure 7.17 uses the (*backupmain*) option to specify that the newly started node (i.e. Main-Container-2) is a third main container, exporting a

## 7.4 Reliability

---

Service Manager address on host host3, port 1099. This new node is to join an existing platform that is specified by the *(-host)*, *(-port)* and *(name)* options.

```
host3: java jade.Boot -backupmain -local-port 1099 -host host1 -port 1099 -name RDADeLE
-services jade.core.replication.MainReplicationService;
jade.core.replication.AddressNotificationService
```

**Figure 7.17:** Generating Replication of Main-Container Producing Main-Container-2

### 7.4.1 Discussion

- To make the RDADeLE system more reliable we suggest that each regional grid has a main container. The main container of each regional grid connects to many non-main containers. These non-main containers represent institution nodes within each administrative region. In turn each node as a non-container, within each administrative region, has many agents which represent learners. This scenario is one of our suggestions which can be adopted in the future.
- Replica catalogues which are provided by the grid middleware could be useful for the agent layer to maintain storage nodes that the administrator configured. Replica catalogues update their information regarding data resources as to when mobility occurs and it maps logical file names to physical locations on grid resources. A resource broker contacts a replica catalogue to query information about data locations.
- All other non-main containers should connect to the replicated main container of its respective regional grid. This is to complete the process of replication.
- Using one or both techniques, **the XRT** and **the RST**, in the prototype does not affect the reliability of the RDADeLE system.



## 7.5 Efficiency in Searching for Learning Objects' Information and Grid Services

### 7.5 Efficiency in Searching for Learning Objects' Information and Grid Services

To measure time intervals the method *System.currentTimeMillis()* is used. The precision of its return value depends on many factors, and to obtain more precise values we conducted the experiment ten times to calculate the mean search time.

#### 7.5.1 Searching for Learning Objects' Information

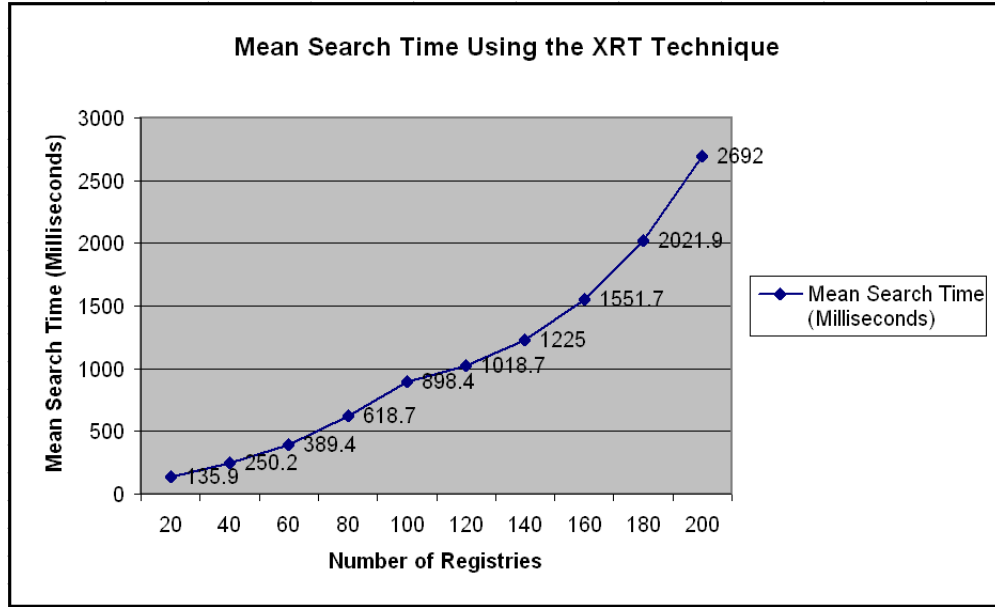
We have tested the RDADeLE prototype against a number of registries which contain meta-data of LOs assuming that there is 1 node and 1 learner. Each administrative region of the Kingdom of Saudi Arabia has only one registry which contains information about LOs in that administrative region. The test has been conducted 10 times for each specific number of registries. The system returns the error (java.lang.OutOfMemoryError: Java heap space) when the number of registries exceeds 200. Table 7.4 shows the mean search time on a different numbers of registries. The first column in the table is the number of registries where the search command performs a search process and the second column is the mean search time in milliseconds.

**Table 7.4:** Mean Search Time for Different Numbers of Registries Using the XRT Technique

Number of Registries	Mean Search Time (Milliseconds)
20	135.9
40	250.2
60	389.4
80	618.7
100	898.4
120	1018.7
140	1225
160	1551.7
180	2021.9
200	2692

## 7.5 Efficiency in Searching for Learning Objects' Information and Grid Services

The graph in figure 7.18 shows the relationship between mean search time (in milliseconds) and numbers of registries using **the XRT technique**. The graph shows that the mean search time for LOs' information is increased with a increase in the number of registries of administrative regions.



**Figure 7.18:** Number of Registries Against Mean Search Time Using the XRT Technique

### 7.5.2 Searching for Registered-based Grid Services

We have tested the RDADeLE prototype against a number of registered grid services assuming that there is 1 regional grid, 1 node and 1 learner. All grid services are registered within the DF which is located in the main container in the Riyadh administrative region. The test has been conducted 10 times for each specific number of registered grid services. The system returns the error (java.lang.OutOfMemoryError: Java heap space) when the number of registered grid services exceeds 4000. Table 7.5 shows the mean search time for different numbers of registered services. The first column in the table is the number of registered services where the search com-

## 7.5 Efficiency in Searching for Learning Objects' Information and Grid Services

---

**Table 7.5:** Mean Search Time for Different Registered Services Using the RST Technique

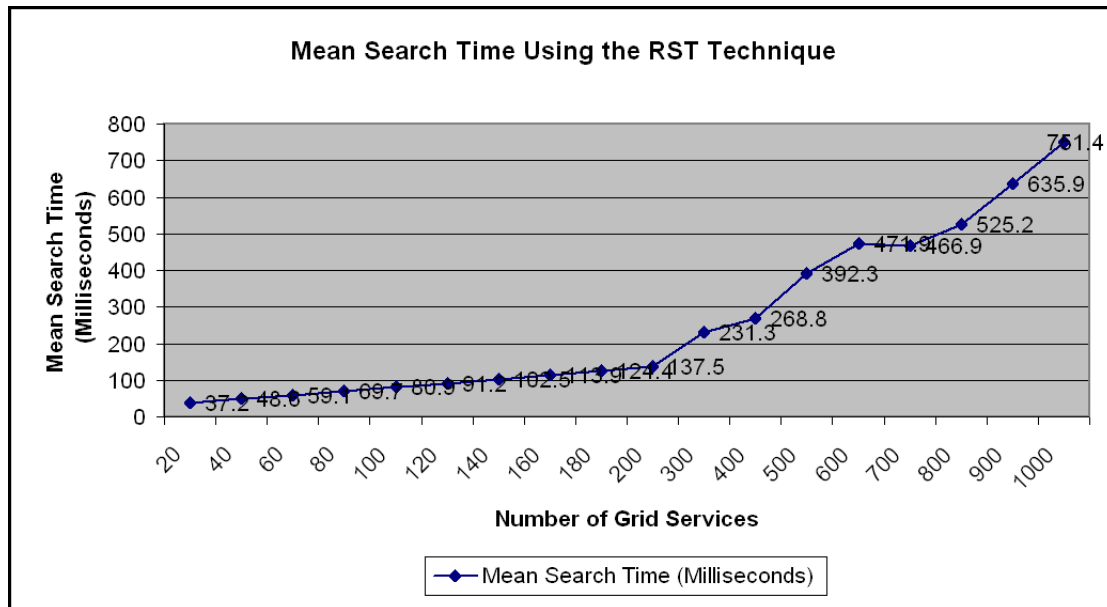
Number of Grid Services	Mean Search Time (Milliseconds)
20	37.2
40	48.6
60	59.1
80	69.7
100	80.9
120	91.2
140	102.5
160	113.9
180	124.4
200	137.5
300	231.3
400	268.8
500	392.3
600	471.9
700	466.9
800	525.2
900	635.9
1000	751.4

mand performs a search process and the second column is the mean search time in milliseconds.

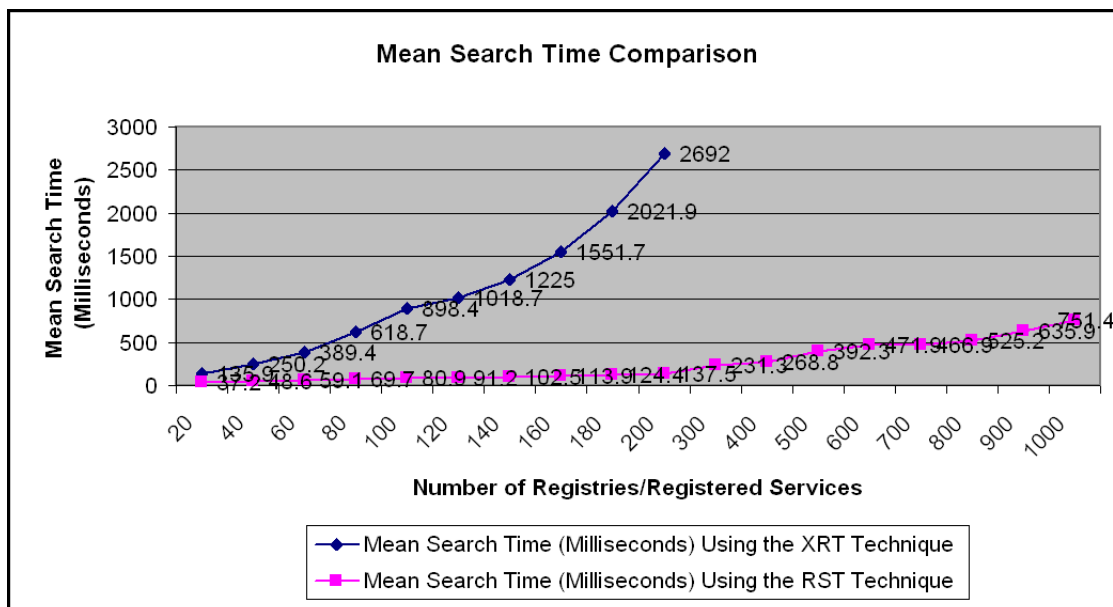
The graph in figure 7.19 shows the relationship between mean search time (in milliseconds) and number of registered services using **the RST technique**. The graph shows that with an increase in the number of grid services registered with the DF there is a corresponding increase in mean search time for grid services..

The graph in figure 7.20 shows the difference in mean search time using both **the XRT** and **the RST** techniques. It shows that more grid services can be registered in the RDADeLE system with less mean search time using **the RST technique**. The search performance increases using **the RST technique** compared with **the XRT technique**.

## 7.5 Efficiency in Searching for Learning Objects' Information and Grid Services



**Figure 7.19:** Number of Registered Services Against Mean Search Time Using the RST Technique



**Figure 7.20:** Mean Search Time Comparison

## 7.6 Review of RDADeLE implementation result towards RDADeLE Requirements

---

### 7.5.3 Discussion

In order to reduce the limitations of the RDADeLE system we suggest the following:

- To avoid low performance or even system crashes we have to avoid setting the variable *maxResult* of the DF search constraint to large values (This is applicable to **the RST technique**). Instead we recommend a distributed platform to activate several containers on different hosts and spread agents across them.
- We can merge both techniques, **XRT** and **RST**, in the prototype and transform some registries into registered services. The transformation reduces the number of registries, which take less time in searching for grid services.
- The graph in figure 7.20 shows the difference in mean search time using both techniques, **XRT** and **RST**. It shows that more grid services could be registered in the RDADeLE system with less mean search time using **the RST technique**. The search performance increases using **the RST technique** compared with **the XRT technique**.

## 7.6 Review of RDADeLE implementation result towards RDADeLE Requirements

This chapter covered the analysis of results to evaluate the RDADeLE architecture. Experiments have been conducted against some factors using both techniques XRT and RST. The second requirement is the extensibility. The result of conducting experiments showed that using the RST technique the RDADeLE environment can be built with more regional grids with less memory consumption. Meanwhile an algorithm has been developed and implemented to provide the RDADeLE architecture

## 7.7 Summary

---

the scalability it needs. The aim of the algorithm is to divide a regional grid into sub-regional grid in case of the increase of connected components to that regional grid. By dividing required regional grids into sub-regional grids we come up with a scalable system of systems.

The second requirement is the flexibility. The result of conducting experiments showed that more grid services could be registered in the RDADeLE system with less mean search time using the RST technique. The search performance increases using the RST technique compared with the XRT technique.

The third requirement is the fault tolerance. In this chapter, we have reviewed and discussed reliability of the RDADeLE architecture. Fault tolerance with cross-notification mechanisms in the RDADeLE architecture provides failure detection and failure handling. Keeping track of and connections between entities of a system plays a big role in the system coherence and control. Replication of the RDADeLE architecture Information Service and the Main Container guarantee the connections between regional grids with each other and connections between components of each regional grid.

## 7.7 Summary

This chapter has provided the simulation for the RDADeLE system to evaluate our system scalability against number of regional grids and number of grid services with simulation results. Afterwards, reliability of the RDADeLE system has been presented with results and discussion. Finally, efficiency of the search process has been evaluated with results and discussion.

# Chapter 8

## Conclusion

### 8.1 Summary

The work presented in this thesis provides an insight into the world of building a dynamic architecture for an e-learning environment which is composed of both grid technology and agent computing. The current description of includes state of the art languages, middleware, and platforms which are used in grid and agent computing.

e-Learning systems are examples of large, complex systems, and systems of systems. Building robust and dynamic systems for e-learning environments require cooperation between many parties. Grid and agent technologies are one of the best combinations to build robust and dynamic systems for e-learning environments. Grid computing is a specialised instance of a distributed system. It is characterised by geographically diverse computer and data resources, these resources are under the control of different administrative domains with different policies and levels of security, and a heterogeneous hardware resource base consisting of components such as processing elements, storage and network. Grid computing is highly dynamic in nature. Whereas, agent technology has many features and characteristics that encourage us to adopt it in the RDADeLE architecture. Characteristics include autonomy, adaptability and proactivity, mobility, persistency, goal orientation, communication, collaboration, co-operation, and flexibility.

In this research, we have developed a dynamic and distributed architecture for an e-learning environment based on both grid and agent technologies. This architecture exploits features of both data grid and agent technologies to create an architecture

## 8.2 Contributions

---

which is more manageable, extensible, flexible and fault tolerant. Scalable data resources of the architecture are built and managed based on data grid technology. The entities of the architecture (e.g. nodes, learners, services) are built and controlled based on agent technology to become flexible and fault tolerant. This distributed architecture allows regional grids to connect to each other, and allows their resources to communicate with each other in order to exchange data between such resources and the search for LOs' information and grid services.

## 8.2 Contributions

The RDADeLE architecture requirements are management, extensibility, flexibility and fault tolerance. The RDADeLE architecture which address these requirements has produced a number of new contributions. These contributions include:

1. The RDADeLE architecture has been built for the purpose of controlling and monitoring regional grids applicable for e-learning systems. We consider these regional grids as systems, which could have subsystems. The RDADeLE architecture controls all elements of these systems using agent technology. In our e-learning environment, a large number of entities (e.g. nodes, resources, learners) connect and disconnect to it at anytime. Using agents in the RDADeLE architecture enriches it to be dynamic so that they control and manage e-learning entities. These agents can easily connect and disconnect from the RDADeLE architecture. The novel contribution here is producing a regional distributed architecture for dynamic e-learning environment that integrates data grid technology with agent technology.
2. Data grid (data resources) has been divided into many grid segments. These grid segments, named regional grids, are distributed. The benefit of division



## 8.2 Contributions

---

is applicable to any dynamic environment. The division in the RDADeLE architecture is not for division itself, but to minimise problems of the whole project (data grid as distributed) by determining and solving problems with each part or segment. Besides that, division is provided to build autonomous regional grids, especially in dynamic environments such as e-learning environments. The division has been adopted in RDADeLE in order to build a system of systems. Eventually, RDADeLE will combine these regional grids to build the whole system. The novel contribution here is dividing the whole data grid into regional grids to compose system of systems environment.

3. Management of data resources is the first requirement in the RDADeLE architecture. Management of data resources in the RDADeLE architecture is performed using data grid middleware which guarantees sharing data resources among users. Part of the management is monitoring the RDADeLE components which performed using administrative agent. The administrative agent is the first type of agents in the RDADeLE architecture. The administrative agent, along with AMS and DF, work as an information service for the RDADeLE system. The administrative agent controls other agents and monitors the RDADeLE system. Moreover, the administrative agent assists the whole RDADeLE system to accomplish its activities by controlling and managing regional agents. The activities of the administrative agent include, for example, creating regional agents (regional grids), authenticating regional grids that connect to the RDADeLE system, registering and/or deregistering regional grids upon request from the regional agent, and listing all registered regional grids upon request from learners. The novel contribution here is to control and manage components of the RDADeLE architecture using developed administrative, regional and node agents.

## 8.2 Contributions

---

4. Extensibility of data resources in the RDADeLE architecture is the second requirement in the RDADeLE architecture. Extensibility refers to scalability of the RDADeLE architecture. On one hand, using grid middleware enriches the RDADeLE architecture to facilitate large-scale data-intensive computing and to provide for collaboration in a wide variety of disciplines. Regional agent and node agent (the second and the third types of agent), besides their functions in controlling regional grids and nodes respectively, cooperate with administrative agent to work intelligently in controlling and monitoring scalable and dynamic e-learning environments. The RDADeLE architecture adapts itself according to the environment scalability to become expandable. Thus the RDADeLE architecture reorganises itself to be able to control and monitor scalable and dynamic environments which are systems of systems. The novel contribution here is to provide the extensibility to the RDADeLE architecture using developed administrative agent.
5. Flexibility: Service agents (the fourth type of agent in the RDADeLE architecture) with other types of agent enrich the architecture with flexibility in composing and searching for LOs' information and services. The result of using agents is producing two techniques in the RDADeLE architecture which have been implemented. The prototype of the RDADeLE architecture simulates two techniques of composing and searching for LOs' information and grid services. The first technique is the XML-based Registries Technique (XRT). Learning objects' information in this technique is built using XML meta-data registries. The second technique is the Registered-based Services Technique (RST). Grid services in this technique, which may have learning objects' information, are built using agents that are registered in order to be discovered and accessed. The novel contribution here is to provide the flexibility to the RDADeLE ar-

### 8.3 Future Work

---

chitecture using developed agents.

6. Fault tolerance: The RDADeLE architecture, as an e-learning system, needs to bind all administrative regions of the Kingdom of Saudi Arabia with each other and with their components in order to maintain reliability to become more robust and reliable. Choosing an appropriate agent platform to assist the architecture to become more reliable and robust is strongly recommended. Using the JADE agent platform to generate the required agents in the RDADeLE architecture is recommended in order to monitor the environment agents and then help the RDADeLE architecture to become more flexible and reliable. The novel contribution here is to provide the fault tolerance to the RDADeLE architecture using the JADE agent platform.

### 8.3 Future Work

There are many ways to build on the work presented in this thesis. The most obvious are as follows:

- Generalising the RDADeLE system and extending it to include many different distributed, large-scale, open, heterogeneous and dynamic applications. These applications include e-banking, e-commerce, and e-health. Such applications have to deal with a large number of members which requires the application to be distributed, large-scale, heterogeneous, and dynamic. Agent technology plays a major role in these kinds of applications. Agents could provide a significant enhancement to these applications to make them more reliable, dynamic, and scalable.
- Improve the flexibility in the RDADeLE architecture by merging both techniques, XRT and RST, in the prototype and transform some registries into

### 8.3 Future Work

---

registered services. The transformation reduces the number of registries, which take less time in searching for grid services.

- Extend the RDADeLE system to use mobile agents in its activities. Mobility and LEAP is useful when using small devices (e.g. Personal Digital Assistant (PDA) and mobile phones) to connect to the RDADeLE system. Small devices can use a browser to access the RDADeLE system to participate as one of the system entities.
- Extend and improve the regional grid policy in the RDADeLE system. Many policy applications could be applied such as are data flow policy, security policy and rule-based policy which could be adopted to control the RDADeLE system.
- Extend and improve the search method for LOs' information and grid services. The search method can adopt an optimal search algorithm in order to improve search time for both LOs' information and grid services.

## Bibliography

- [1] Ministry of Foreign Affairs, Saudi Arabia, 2009. <http://www.mofa.gov.sa/>.
- [2] Sharing Infrastructure and Resources in Europe (Sirene), Accessed January 2010. <http://www.eugrid.eu/home>.
- [3] David Abramson, Rajkumar Buyya, and Jonathan Giddy. A computational economy for grid computing and its implementation in the nimrod-G resource broker. *Future Generation Comp. Syst*, 18(8):1061–1074, 2002.
- [4] Ali M. Al-Asmari. *The Use Of The Internet Among EFL Teachers At The Colleges Of Technology In Saudi Arabia*. PhD thesis, The Ohio State University, 2005.
- [5] Ahmed Al-Nazer and Tarek Helmy. A web searching guide: Internet search engines & autonomous interface agents collaboration. In *Web Intelligence/IAT Workshops*, pages 424–428. IEEE, 2007.
- [6] H. Al-Sakran. An agent-based architecture for developing e-learning system. *Information Technology Journal*, 5(1):121–127, 2006.
- [7] Omar Subhi Aldabbas. *A Framework for Mobility and Temporal Dimensions of Grid Systems*. PhD thesis, De Montfort University, 2009.
- [8] E. AlHashel, B.M. Balachandran, and D. Sharma. Extending prometheus with agent cooperation. In *Computational Intelligence for Modelling Control & Automation, 2008 International Conference on*, pages 912–918, Dec. 2008.
- [9] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke. Gridftp: Protocol extensions to ftp for the grid. February 27 2001.
- [10] S. AlZahrani, A. Ayesh, and H. Zedan. Multi-agent Based Dynamic e-Learning Environment. *International Journal of Information Technology and Web Engineering (IJITWE)*, 4(2), June 2008.
- [11] S. AlZahrani, A. Ayesh, and H. Zedan. Multi-agent System Based Regional Data Grid. In *Computer Engineering & Systems, 2008. ICCES 2008. International Conference on*, pages 337–342, Nov. 2008.
- [12] S. AlZahrani, A. Ayesh, and H. Zedan. Regionally Distributed Architecture for Dynamic e-Learning Environment (RDADeLE). In *Human System Interactions, 2008 Conference on*, pages 579–584, May 2008.
- [13] Rachid Anane, Kuo-Ming Chao, and Yinsheng Li. Hybrid composition of web services and grid services. In *EEE*, pages 426–431. IEEE Computer Society, 2005.

## BIBLIOGRAPHY

---

- [14] John Langshaw Austin. *How to Do Things with Words*. Harvard University Press, 1962.
- [15] Kentaro Fukui Bart Jacob, Michael Brown and Nihar Trivedi. Introduction to grid computing. Technical Report 1, IBM International Technical Support Organisation, 2005.
- [16] Norbert Bieberstein Candice Gilzean Jean-Yves Girard Roman Strachowski Bart Jacob, Luis Ferreira and Seong (Steve) Yu. Enabling applications for grid computing with globus. Technical report, IBM International Technical Support Organisation, 2003.
- [17] Sujoy Basu, Sujata Banerjee, Puneet Sharma, and Sung-Ju Lee. Nodewiz: peer-to-peer resource discovery for grids. In *CCGRID*, pages 213–220. IEEE Computer Society, 2005.
- [18] Bernhard Bauer, Jörg P. Müller, and James Odell. Agent UML: A formalism for specifying multiagent software systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):91–104, 2001.
- [19] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, NJ, April 2007.
- [20] Federico Bergenti and Agostino Poggi. Supporting agent-oriented modelling with UML. *International Journal of Software Engineering and Knowledge Engineering*, 12(6):605–618, 2002.
- [21] Carole Bernon, Marie-Pierre Gleizes, Gauthier Picard, and Pierre Glize. The ADELFE methodology for an intranet system design. In P Giorgini, Y Lespérance, G Wagner, and E Yu, editors, *International Bi-Conferenystems (AOIS-2002) at CAice Workshop on Agent-Oriented Information SSE'02 (AOIS - SSE), Toronto, Ontario, Canada, 27/05/02-28/05/02*, page (on line), <http://ceur-ws.org>, mai 2002. CEUR Workshop Proceedings.
- [22] Joseph P. BIGUS and Jennifer BIGUS. *Constructing intelligent agents using java*. Wiley, 2001. second edition.
- [23] Jeffrey M. Bradshaw. An introduction to software agents. In Jeffrey M. Bradshaw, editor, *Software Agents*, chapter 1, pages 3–46. AAAI Press / The MIT Press, 1997.
- [24] Frances M. T. Brazier, Catholijn M. Jonker, and Jan Treur. Principles of component-based design of intelligent agents. *Data and Knowledge Engineering*, 41(1):1–27, 2002.

## BIBLIOGRAPHY

---

- [25] Sabin-Corneliu Buraga. Developing agent-oriented e-learning systems. In *in Proceedings of The 14th International Conference on Control Systems And Computer Science vol. II, I. Dumitrache and C. Buiu, Eds, Politehnica*. Politehnica Press, Bucharest, June 16 2003.
- [26] P Busetta, R Rönquist, A Hodgson, and A Lucas. JACK intelligent agents - components for intelligent agents in java. Technical report, European Coordination Action for Agent Based Computing(AgentLink), January 1999.
- [27] Rajkumar Buyya and M. Manzur Murshed. Gridsim: A toolkit for the modeling and simulation of Distributed Resource Management and Scheduling for Grid Computing. *CoRR*, cs.DC/0203019, 2002. informal publication.
- [28] Higher Education Statistics Center. Statistical report 2009. Statistical, Ministry of Higher Education, Saudi Arabia, 2009. <http://statistics.mohe.gov.sa/>.
- [29] Luca Cernuzzi, Massimo Cossentino, and Franco Zambonelli. Process models for agent-based development. *Journal of Engineering Applications of Artificial Intelligence*, 18(2):205–222, 2005.
- [30] Soumyakanti Chakraborty, Anup K. Sen, and Amitava Bagchi. Designing proxy bidders for online combinatorial auctions. In *HICSS*, pages 1–10. IEEE Computer Society, 2009.
- [31] Mohan Baruwat Chhetri, Rosanne Price, Shonali Krishnaswamy, and Seng Wai Loke. Ontology-based agent mobility modelling. In *HICSS*. IEEE Computer Society, January 2006.
- [32] Krzysztof Chmiel, Maciej Gawinecki, Pawel Kaczmarek, Michal Szymczak, and Marcin Paprzycki. Efficiency of JADE agent platform. *Scientific Programming*, 13(2):159–172, 2005.
- [33] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. <http://www.w3.org/TR/wsdl>, 2001.
- [34] Li Chunlin and Li Layuan. Agent framework to support the computational grid. *The Journal of Systems and Software*, 70(1–2):177–187, February 2004.
- [35] Andrea Clematis, Paola Forcheri, and Alfonso Quarati. Interacting with learning objects in a distributed environment. In *PDP*, pages 322–329. IEEE Computer Society, 2006.
- [36] Luc Clement, Andrew Hatley, Claus von Riegen, and Tony Rogers. UDDI version 3.0.2. Technical report, Organization for the Advancement of Structured Information Standards, UDDI Spec Technical Committee Draft, October 2004.

## BIBLIOGRAPHY

---

- [37] The IEEE Learning Technology Standards Committee. The learning object metadata standard. Technical report, The IEEE, 2007.
- [38] World Wide Web Consortium. Web services architecture, 2004. <http://www.w3.org/TR/ws-arch/>.
- [39] Marco Cremonini, Andrea Omicini, and Franco Zambonelli. Building mobile agent applications in hiMAT. In *PDSE*, pages 174–181, 1999.
- [40] Scott A. DeLoach, Mark F. Wood, and Clint H. Sparkman. Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):231–258, 2001.
- [41] Gizella Dewath. A study of the current state of e-learning in the united kingdom. Technical report, The International Dunhuang Project The British Library, May 2004.
- [42] Kevin R. Dixon, Theodore Q. Pham, and Pradeep K. Khosla. Port-based adaptable agent architecture. *Lecture Notes in Computer Science*, 1936:181, 2001.
- [43] Dirk Düllmann, Wolfgang Hoschek, Francisco Javier Jaén-Martínez, Ben Segal, Heinz Stockinger, Kurt Stockinger, and Asad Samar. Models for replica synchronisation and consistency in a data grid. In *HPDC*, pages 67–75. IEEE Computer Society, 2001.
- [44] Agostino Poggi Fabio Bellifemine, Giovanni Caire and Giovanni Rimassa. Jade: A white paper. *EXP in search of innovation*, September 2003.
- [45] Tiziana Trucco Giovanni Rimassa Roland Mungenast Fabio Bellifemine, Giovanni Caire. Jade administrators guide. Technical report, Telecom Italia, February 2006.
- [46] M. Fasli. *Agent Technology for E-commerce*. Wiley & Sons, 2007.
- [47] J. Ferber. *Multi-agent Systems: Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.
- [48] Roy T. Fielding, Jim Gettys, Jeff Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext Transfer Protocol — HTTP/1.1. Internet Request for Comment RFC 2616, Internet Engineering Task Force, June 1999.
- [49] Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In Jeffrey M. Bradshaw, editor, *Software Agents*, chapter 14, pages 291–316. AAAI Press / The MIT Press, 1997.



## BIBLIOGRAPHY

---

- [50] Marco Fioretti. The OASIS standard for office documents. *Linux Journal*, 2004(119):4–4, March 2004.
- [51] M. Fisher. A survey of Concurrent METATEM — the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Temporal Logic — Proceedings of the First International Conference (LNAI Volume 827)*, pages 480–505. Springer-Verlag: Heidelberg, Germany, July 1994.
- [52] The Foundation for Intelligent Physical Agents (FIPA). Fipa acl message structure specification. Technical Report SC00061G, The Foundation for Intelligent Physical Agents (FIPA), December 2002. <http://www.fipa.org/specs/fipa00061/SC00061G.html>.
- [53] The Foundation for Intelligent Physical Agents (FIPA). Major implementations of agent platforms which conform to the fipa specifications. Technical report, The Foundation for Intelligent Physical Agents (FIPA), 2003. <http://www.fipa.org/resources/livesystems.html>.
- [54] A. Savva D. Berry A. Djaoui A. Grimshaw B. Horn F. Maciel F. Siebenlist R. Subramaniam J. Treadwell J. Von Reich Foster, H. Kishimoto. The open grid services architecture. Technical Report 1, Global Grid Forum (GGF), January 2005.
- [55] I. Foster and C. Kesselman, editors. *The Grid2: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, 2004.
- [56] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 28 2002.
- [57] Ian T. Foster, Nicholas R. Jennings, and Carl Kesselman. Brain meets brawn: Why grid and agents need each other. In *AAMAS*, pages 8–15. IEEE Computer Society, 2004.
- [58] Ian T. Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *IJHPCA*, 15(3):200–222, 2001.
- [59] Angelo Gaeta, Pierluigi Ritrovato, Francesco Orciuoli, and Matteo Gaeta. Enabling technologies for future learning scenarios: the semantic grid for human learning. In *CCGRID*, pages 3–10. IEEE Computer Society, 2005.
- [60] Youssef Gamha, Nacéra Bennacer, Guy Vidal-Naquet, Béchir el Ayeb, and Lotfi Ben Romdhane. A framework for the semantic composition of web services handling user constraints. In *ICWS*, pages 228–237. IEEE Computer Society, 2008.

- [61] Alfredo Garro, Luigi Palopoli, and Francesco Ricca. Exploiting agents in e-learning and skills management context. *AI Commun*, 19(2):137–154, 2006.
- [62] José Manuel Gascueña and Antonio Fernández-Caballero. Prometheus and INGENIAS agent methodologies: A complementary approach. In Michael Luck and Jorge J. Gómez-Sanz, editors, *AOSE*, volume 5386 of *Lecture Notes in Computer Science*, pages 131–144. Springer, 2008.
- [63] M. R. Genesreth, R. E. Fikes, et al. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.
- [64] Paolo Giorgini, John Mylopoulos, and Roberto Sebastiani. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Engineering Applications of Artificial Intelligence*, 18(2):159–171, 2005.
- [65] Andrew S. Grimshaw, Marty A. Humphrey, and Anand Natrajan. A philosophical and technical comparison of legion and globus. *IBM Journal of Research and Development*, 48(2):233–254, 2004.
- [66] Martin Gudgin, Marc Hadley, and Tony Rogers. Web services addressing 1.0 - core. W3C recommendation, W3C, May 2006. <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509>.
- [67] Kasper Hallenborg, Ask Just Jensen, and Yves Demazeau. Reactive agent mechanisms for manufacturing process control. In *Web Intelligence/IAT Workshops*, pages 399–403. IEEE, 2007.
- [68] A Hamdan. Women and education in saudi arabia: Challenges and achievements. *International Education Journal*, 6(1):42–64, 2005.
- [69] S. Hammami, H. Mathkour, and E.A. Al-Mosallam. A multi-agent architecture for adaptive e-learning systems using a blackboard agent. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 184–188, Aug. 2009.
- [70] Khadidja Harbouche and Mahieddine Djoudi. Agent-based design for elearning environment. *Journal of Computer Science*, 3(6):383–389, 2007.
- [71] S. Hasegawa and K. Ochimizu. A learning management system based on the life cycle management model of e-learning courseware. In *Advanced Learning Technologies, 2005. ICAALT 2005. Fifth IEEE International Conference on*, pages 35–37, July 2005.
- [72] Fu-Shiung Hsieh. Holarchy formation and optimization in holonic manufacturing systems with contract net. *Automatica*, 44(4):959–970, 2008.

## BIBLIOGRAPHY

---

- [73] Fu-Min Huang and Ming Chao. An architecture of virtual environment for E-learning (AVEE). In *ICALT*, pages 148–149. IEEE Computer Society, 2005.
- [74] Wei Huang, Elia El-Darzi, and Li Jin. Extending the gaia methodology for the design and development of agent-based software systems. In *COMPSAC*, pages 159–168. IEEE Computer Society, 2007.
- [75] Summer Scott Huyette. *Political adaptation in Saudi Arabia a study of the Council of Ministers*. PhD thesis, Columbia University, 1984.
- [76] Francisco Curbera Ibm, Hitesh Dholakia, Yaron Goland Bea, Johannes Klein Microsoft, Frank Leymann Ibm, Kevin Liu Sap, Dieter Roller Ibm, Doug Smith, Siebel Systems, Satish Thatte, Ivana Trickovic Sap, and Sanjiva Weerawarana Ibm. Business process execution language for web services, May 13 2003.
- [77] C. A. Iglesias, M. Garijo, J. C. Gonzalez, and J. R. Velasco. Analysis and design of multiagent systems using MAS-CommonKADS. *Lecture Notes in Computer Science*, 1365:313, 1998.
- [78] N. R. Jennings and M. Wooldridge. Applying agent technology. *Applied Artificial Intelligence*, 9(6):357–370, 1995.
- [79] Hai Jin and Li Qi. Agents in chinagrid support platform. In *Cluster Computing and the Grid Workshops CCGRID, Sixth IEEE International Symposium*, volume 2, pages 1–4. IEEE Computer Society, May 2006.
- [80] L.F. Johnson. Elusive vision: Challenges impeding the learning object economy. White paper, Macromedia, 2003.
- [81] Thomas Juan, Adrian R. Pearce, and Leon Sterling. ROADMAP: extending the gaia methodology for complex open systems. In *AAMAS*, pages 3–10. ACM, 2002.
- [82] L. P. Kaelbling. A situated automata approach to the design of embedded agents. *SIGART Bulletin*, 2(4):85–88, 1991.
- [83] Do-Hyeon Kim and Kyung-Woo Kang. Design and implementation of integrated information system for monitoring resources in grid computing. In *CSCWD*, pages 1–6. IEEE, 2006.
- [84] Weining Kong, Junzhou Luo, and Tiantian Zhang. A workflow based E-learning architecture in service environment. In *CIT*, pages 1026–1032. IEEE Computer Society, 2005.
- [85] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper*, 32(2):135–164, 2002.

## BIBLIOGRAPHY

---

- [86] Profactor & Telecom Italia Lab. Analysis and benchmark of scalability and performance of jades. Technical report, Telecom Italia, November 2004.
- [87] John Laird and Michael Van Lent. Human-level AI's killer application: Interactive computer games. *The AI Magazine*, 22(2):15–25, 2000.
- [88] Danny B. Lange and Daniel T. Chang. Programming mobile agents in java – A white paper. Technical report, IBM Corp, 1996.
- [89] Lorenzo Lazzari, Marco Mari, and Agostino Poggi. Cafe - collaborative agents for filtering e-mails. In *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 356–361, Washington, DC, USA, 2005. IEEE Computer Society.
- [90] Jaewook Lee. *Design Collaboration as a Framework for Building Intelligent Environments*. PhD thesis, University of Illinois, 2006.
- [91] Arnaud Legrand, Loris Marchal, and Henri Casanova. Scheduling distributed applications: the simgrid simulation framework. In *CCGRID*, pages 138–145. IEEE Computer Society, 2003.
- [92] I. Legrand, H. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, C. Dobre, A. Muraru, A. Costan, M. Dediu, and C. Stratan. Monalisa: An agent based, dynamic service system to monitor, control and optimize distributed systems. *Computer Physics Communications*, 180(12):2472 – 2498, 2009.
- [93] Frank Leymann. Web services flow language (WSFL 1.0). Technical report, IBM, May 2001.
- [94] Bingchen Li, Wei Li, and Zhiwei Xu. Implementation issues of a grid service markup language. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 620–624, Aug. 2003.
- [95] Wang Li, Wang Cong, Long Hao, and Di Rui-Hua. An Adaptive MAS-based Data Acquisition Model in ESESGrid. In *ChinaGrid Annual Conference, 2008. ChinaGrid '08. The Third*, pages 218–222, Aug. 2008.
- [96] Shu-Sheng Liaw. Investigating students' perceived satisfaction, behavioral intention, and effectiveness of e-learning: A case study of the blackboard system. *Computers & Education*, 51(2):864–873, 2008.
- [97] Michael Luck, Peter McBurney, and Chris Preist. *Agent Technology: Enabling next generation computing: a roadmap for agent based computing*. Agentlink, 2003.

## BIBLIOGRAPHY

---

- [98] P. Maes. The agent network architecture (ANA). *SIGART Bulletin*, 2(4):115–120, 1991.
- [99] Muthucumaru Maheswaran and Klaus Krauter. A parameter-based approach to resource discovery in grid computing system. In Rajkumar Buyya and Mark Baker, editors, *GRID*, volume 1971 of *Lecture Notes in Computer Science*, pages 181–190. Springer, 2000.
- [100] C. E. Martin, K. S. Barber, and K. S. Barber. Agent autonomy: Specification, measurement, and dynamic adjustment. In *In Proceedings of the Autonomy Control Software Workshop, Agents &apos;99*, pages 8–15, 1999.
- [101] Marilyn McClelland. Metadata standards for educational resources. *IEEE Computer*, 36(11):107–109, 2003.
- [102] Nilo Mitra and Yves Lafon. SOAP version 1.2 part 0: Primer (second edition). World Wide Web Consortium, Recommendation REC-soap12-part0-20070427, April 2007.
- [103] Mitsubishi Electric Information Technology Center America. Concordia, mobile agent computing. A White Paper, January 1998. <http://www.cis.upenn.edu/bcpierce/courses/629/papers/Concordia-WhitePaper.html>.
- [104] Kazuo Miyashita and Gautam Rajesh. Multiagent coordination for controlling complex and unstable manufacturing processes. *Expert Systems with Applications*, 37(3):1836 – 1845, 2010.
- [105] Paul Mockapetris. Domain names concepts and facilities. Technical report, Information Sciences Institute, University of Southern California, November 1987.
- [106] L. Monostori, J. Vncza, and S.R.T. Kumara. Agent-based systems for manufacturing. *CIRP Annals - Manufacturing Technology*, 55(2):697 – 720, 2006.
- [107] Elena Nardini, Andrea Omicini, and Mirko Viroli. General-purpose coordination abstractions for managing interaction in MAS. In *Web Intelligence/IAT Workshops*, pages 501–506. IEEE, 2009.
- [108] United Nations. World population prospects the 2006 revision. Technical report, Division of Economic and Social Affairs, 2006.
- [109] Patricia Albanese (Pitkin) Bradley F. Baker David Cohen Lorcan Dempsey Neil McLean, Heidi Sander et al. Libraries and the enhancement of e-learning. *OCLC E-Learning Task Force*, 43017-3395, October 2003.

## BIBLIOGRAPHY

---

- [110] Z. Németh and V. Sunderam. A formal framework for defining grid systems. In *Proceedings of the Second IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID2002*, pages 202–211. IEEE Computer Society Press, May 2002.  
ASMs are used to define a model for grid systems.
- [111] Eric Newcomer. *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [112] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, 1990.
- [113] Rob Philpott Nick Ragouzis, John Hughes and Eve Maler. Security assertion markup language (SAML) v2.0. Technical Report 2, OASIS Standard, October 2006.
- [114] Hyacinth S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(3):1–40, Septemebr 1996.
- [115] Object Management Group. Agent technology — green paper. OMG Document agent/00-09-01, September 2000.
- [116] G. O’Hare and N. Jennings. *Foundations of Distributed Artificial Intelligence*. Wiley, 1996.
- [117] Lin Padgham and Michael Winikoff. Prometheus: A pragmatic methodology for engineering intelligent agents. In *In Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies*, pages 97–108, October 01 2002.
- [118] Victor Pankratius, Olivier S, and Wolffried Stucky. Retrieving content with agents in web service e-learning systems. In *Symposium on Professional Practice in AI, IFIP WG12.5 in Proceedings of the First IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI)*, pages 91–100, 2004.
- [119] E. Pascalau. Smart agent based on rules for web knowledge mining. In *Digital EcoSystems and Technologies Conference, 2007. DEST ’07. Inaugural IEEE-IIES*, pages 458–461, Feb. 2007.
- [120] Loris Penserini, Anna Perini, Angelo Susi, and John Mylopoulos. High variability design for software agents: Extending tropos. *TAAS*, 2(4), 2007.
- [121] J. Postel. Transmission control protocol. Technical Report RFC 793, DARPA, September 1980.
- [122] J. Postel. User datagram protocol. Technical Report RFC 768, Information Sciences Institute, University of Southern California, August 1980.

## BIBLIOGRAPHY

---

- [123] J. Postel. Internet protocol. *Network Information Center RFC 791*, pages 1–45, September 1981.
- [124] Jon Postel. Internet control message protocol. RFC 792, ISI, September 1981.
- [125] J. S. Rosenschein and G. Zlotkin. Designing conventions for automated negotiation. *AI Magazine*, pages 29–46, Fall 1994.
- [126] A. Roy, J.; Ramanujan. Understanding Web services. *IT Professional*, 3(6):Pages:69–73, Nov/Dec 2001.
- [127] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
- [128] I. Foster J. Frey S. Graham C. Kesselman S. Tuecke, K. Czajkowski. Grid service specification. Technical report, Open Grid Services Infrastructure WG, Global Grid Forum, July 11 2002. Draft 3.
- [129] Andrew P. Sage and Christopher D. Cuppan. On the systems engineering and management of systems of systems and federations of systems. *Inf. Knowl. Syst. Manag.*, 2(4):325–345, 2001.
- [130] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, England, 1969.
- [131] Zhidong Shen and Qiang Tong. A security technology for mobile agent system improved by trusted computing platform. In *Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on*, volume 3, pages 46–50, Aug. 2009.
- [132] Hongchi Shi, Spyridon Revithis, and Su-Shing Chen. An agent enabling personalized learning in e-learning environments. In *AAMAS*, pages 847–848. ACM, 2002.
- [133] Yoav Shoham. An overview of agent-oriented programming. In Jeffrey M. Bradshaw, editor, *Software Agents*, chapter 13, pages 271–290. AAAI Press / The MIT Press, 1997.
- [134] Herbert A. Simon. *The New Science of Management Decision*. Harper and Brothers, New York, 1960.
- [135] Thomas Skylogiannis, Grigoris Antoniou, Nick Bassiliades, and Guido Governatori. DR-NEGOTIATE - A system for automated agent negotiation with defeasible logic-based strategies. In *EEE*, pages 44–49. IEEE Computer Society, 2005.
- [136] H. J. Song, X. Liu, D. Jakobsen, et al. The MicroGrid: A scientific tool for modeling Computational Grids. *Scientific Programming*, 8(3):127–141, 2000.

## BIBLIOGRAPHY

---

- [137] Ron Sun. Accounting for the computational basis of consciousness: A connectionist approach. *Consciousness and Cognition*, 8:529–565, September 06 1999.
- [138] Ron Sun. *Duality of The Mind*. Lawrence Erlbaum Associates, Inc., 2002.
- [139] Yong Sun and Bo Wu. Agent hybrid architecture and its decision processes. In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 641–644, Aug. 2006.
- [140] Keith Swenson. Simple workflow access protocol (SWAP). Technical report, 1998. <http://www.ics.uci.edu/ietfswap/>.
- [141] Domenico Talia. The open grid services architecture: Where the grid meets the web. *IEEE Internet Computing*, 6(6):67–71, 2002.
- [142] Technical and Vocational Training Corporation (TVTC). Annual report 2008. Annual, Technical and Vocational Training Corporation (TVTC), Saudi Arabia, 2008. <http://www.tvtc.gov.sa/Downloads/Reports/annualreport.pdf>.
- [143] I. Foster J. Frey S. Graham C. Kesselman T. Maguire T. Sandholm P. Vanderbilt D. Snelling Tuecke, K. Czajkowski. The open grid services infrastructure (OGSI) version 1.0, 2003. [http://www.globus.org/alliance/publications/papers/Final\\_OGSI\\_Specification\\_V1.0.pdf](http://www.globus.org/alliance/publications/papers/Final_OGSI_Specification_V1.0.pdf).
- [144] Steve Tuecke. Grid an option for data management challenges. *Computerworld*, 2006.
- [145] Amund Tveit. A survey of agent-oriented software engineering, July 12 2001.
- [146] Giuseppe Vizzari and Francesco Olivieri. Towards hybrid situated agents based virtual environments. In *Web Intelligence/IAT Workshops*, pages 587–590. IEEE, 2009.
- [147] Yong Wang, Chunming Hu, and Jinpeng Huai. A new grid workflow description language. In *IEEE SCC*, pages 257–260. IEEE Computer Society, 2005.
- [148] Gerhard Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, MA, USA, 1999.
- [149] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, Chichester, 2002.
- [150] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.



## BIBLIOGRAPHY

---

- [151] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. A methodology for agent-oriented analysis and design. In *Proceedings of Agents 1999*, pages 69–76, 1999.
- [152] Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.
- [153] Zheng Xiu-Ying, Chang Gui-Ran, Li Zhen, and Wang Jian. A resource-centric p2p network model for grid resource discovery. In *Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on*, volume 1, pages 210–214, Aug. 2008.

## Appendix A

### Java Code of RDADeLE implementation

The following is the Java source code of the RDADeLE implementation.