A FRAMEWORK FOR THE VISUALISATION AND CONTROL OF UBIQUITOUS DEVICES, SERVICES AND DIGITAL CONTENT

By

Amjad Shaheed BSc (Hons)

A thesis submitted in partial fulfilment of the
Requirements of Liverpool John Moores University
for the degree of Doctor of Philosophy

Networked Appliances Laboratory
School of Computing and Mathematical Sciences
January 2011

THE FOLLOWING HAVE NOT BEEN COPIED ON INSTRUCTION FROM THE UNIVERSITY

Figure 2.2 page 29

Figure 2.3 page 30

Figure 2.4 page 31

Figure 2.6 page 37

This thesis is dedicated to,

My Brother Haroon Shaheed 1978 - 2001

ACKNOWLEDGEMENTS

I would like to take the opportunity to pay thanks to the people who guided and supported me throughout my PhD. Without their support and contribution, successful completion of my research would not have been possible.

I would like to express my gratitude to my supervisor, Professor Madjid Merabti for his continuous support, encouragement, guidance and invaluable suggestions during this research. I would also like to say special thanks to my second supervisor Dr. Paul Fergus for, his numerous helpful discussions and his valuable time to help me with various stages of my project, in particular, giving valuable suggestions and comments on my work. Additionally I am deeply indebted to my first supervisor Dr. Omar Abuelmaatti his sincere guidance and encouragement throughout my research, which has been invaluable and unsparing. Therefore, I do want to show my deep appreciation to him. I want to thank Prof. El Rhalibi, Abdennour for those long discussions and giving me time to review my thesis, which helped me a lot during difficult moments in my studies

My thanks also go to the researchers, administration staff and technicians in the School of Computing and Mathematical Sciences at Liverpool John Moores University for their support over the past years. I would like to show my great appreciation to my beloved parents, brothers and sister for their support and guidance during all these years. I am also thankful to my uncle Malik Rehman for his encouragement, discussions and support throughout my life.

I would like to thanks my friends Dr Kashif Kifayat, Dr Mohammad Arshid, Dr Huma Javeed, Dr Mohammad Asim, Mohammad Haroon for their help and support. My thanks go to Anastasia Minadaki and her family for the support and help during my study.

I would like to special thanks to Guido Ballermann and Thilo Göricke for helping me during my research.

ABSTRACT.

There is undoubtedly an increase in the number of consumer appliances enjoying networking capabilities. With data throughput increasing among devices that are renowned to be less capable than personal computers, we see an increase in the consumption of multimedia. In parallel, gaming and social networking are at the forefront of next generation entertainment systems where new and novel usage scenarios are pushing technological boundaries. These advances have certainly provided a platform for innovation, where a natural progression would be to bridge the gap between the aforementioned technologies. For example, content sharing over networked devices, beyond simple file sharing is becoming a reality.

Furthermore, many devices such as mobile phone are forming closer relationships with different virtual worlds, such as World of Warcraft and Second Life. In one sense, the boundaries between the two are becoming increasingly less distinct. Consequently, this opens up many new avenues for content sharing, not only between devices but also between sophisticated virtual worlds. Given such interoperable platforms, a natural progression would see content that seamlessly resides within either. This will open up new opportunities where third party content providers and users alike are able to create and share content over these new platforms.

Achieving this will require several challenges to be addressed. These include service-oriented networking; behaviour discovery; behaviour capability matching and on the fly behaviour generation in required target environments. Overcoming these challenges will allow mechanisms to be developed that enable us to move content seamlessly between heterogeneous environments (virtual and real).

In this thesis, we describe a novel approach that we have successfully developed to address these challenges – a Framework for the Visualisation and Control of Ubiquitous Devices and Services. Our framework allows us to move easily between real and virtual environments where the content and services we use are always at our disposal. Utilising the benefits of being connected will allow us to manage our content and services independently of where they reside. Our framework provides a basis on which this vision can be realised, that facilitates the sharing of objects across different environments, both virtual and real. This is performed in a flexible way where containers within different real and virtual environments can consume such objects. Using the semantic descriptions of the behaviours objects support, target environments will create scripted behaviours based on interpretations of the functionality they provide. This enables applications, underpinned by digital content, such as dynamic game development, immersive and interactive 3D multimedia, and on-the-fly scene analysis, to emerge.

We have successfully developed a real world prototype that implements our vision of a Smart Home Framework, which we have then used to evaluate our framework. Using our framework, we can discover ubiquitous computing devices, such as mobile phones, represent these devices graphically and generate the behaviour of these devices on the fly. In this scenario, for example, the framework enabled the making of a call, answering an incoming call, and sharing multimedia content with a correspondent avatar in the virtual world mapped to a physical device.

ACKNOWLEDGEMENTS ii
ABSTRACT. I
LIST OF FIGURES VII
LIST OF TABLES X
LIST OF ACRONYMS AND TERMS XI
1 Introduction 1
1.1 Preamble 1
1.2 Ubiquitous Devices 4
1.3 Networked Appliances and Home Networking 5
1.4 Virtual Environment 6
1.5 Incorporating Rule Based System and Dynamic Scripting 7
1.6 Scope of the Research 9
1.7 Project Requirements 9
1.8 Novel Contributions to Knowledge 11
1.8.1 3D Visualisation in Ubiquitous Computing, Device and Services 11
1.8.2 Content Sharing Across Virtual and Physical Environments 12
1.8.3 Object Behaviour Matching 14
1.8.4 Interaction and Control in Ubiquitous Computing System 15
1.9 Thesis Structure 16
2 Virtual Environments, Rules, Dynamic Scripting and Peer-to-Peer Networking 19
2.1 Introduction 19
2.2 Computer Gaming, Entertainment and Social networking 19
2.2.1 Computer Games 19
2.2.2 Social Networking 22
2.3 3D Virtual Environment Technologies 23
2.3.1 3D Modelling 24
2.3.2 Positioning in Space 24
2.3.3 Textures and Colours 25
2.3.4 3D Gaming Engine 26
2.3.5 Scene Graph 27
2.3.6 Collision Detection 27
2.4 Rule-Based Systems 28

2.5 Dynamic Scripting....32
2.6 Peer to Peer Networking....33
2.6.1 P2P Models....35
2.6.2 Name Based Classification....37
2.6.3 P2P Applications....38
2.7 Summary....39
3 Ubiquitous Computing and Networked Appliances....40
3.1 Introduction....40
3.2 Ubiquitous Computing....40
3.2.1 Aware House project....42
3.2.2 Sixth Sense....42
3.3 Networked Appliances....43
3.3.1 Digital Living Network Alliance project (DLNA)....43
3.3.2 Open Services Gateway Initiative (OSGi)....44
3.3.3 Reconfigurable, Ubiquitous, Networked Embedded Systems (RUNES)....45
3.3.4 UPnP....45
3.4 Summary....46
3.4.1 Challenges....46
4 A Framework for the Visualisation and Control of Ubiquitous Devices, Services and Digital Contents....48
4.1 Introduction....48
4.2 Framework Overview....49
4.3 Visual Resource Manager Protocol....54
4.3.1 Protocol Requirements....55
4.3.2 Protocol Overview....56
4.3.3 Protocol Design....56
4.4 Asset Lookup and Resource Monitor Protocol....68
4.4.1 Protocol requirements....68
4.4.2 Protocol Operation....69
4.5 Rule Engine....70
4.5.1 Rule Engine Requirements....71
4.5.2 Rule Engine Design Overview....71

4.6 Script Parser and generator 73
4.6.1 Script Parser and Generator Requirements 73
4.6.2 Script Parser and Generator Operation 73
4.7 Behaviours Ontologies 75
4.7.1 Semantic Matching and Generation of Behaviours 76
4.8 Summary 82
5 Case Study: Smart Home Environment 83
5.1 Introduction 83
5.2 Case Study 83
5.2.1 Characteristics of this study: 91
5.2.2 Using Our Framework for a Smart Home Environment 92
5.2.3 Positive aspects of this Case Study 93
5.3 Other Application Domains 93
5.3.1 Games 93
5.3.2 A 3D Internet Interactive Commerce 94
5.4 Summary 95
6 System Implementation 96
6.1 Introduction 96
6.2 Framework Architecture 96
6.3 Framework Services 96
6.4 Visual Resource Manager 97
6.5 Secondary and Application Specific Services 98
6.5.1 Asset Lookup and Resource Monitor 98
6.5.2 Visualisation Engine 98
6.5.3 Scripting Engine 104
6.5.4 Rule Engine 108
6.5.5 The SUF Peer-to-Peer framework 109
6.5.6 Behaviour Matching Services 110
6.6 The Framework prototype 111
6.6.1 Technical Description 113
6.6.2 Prototype Configuration 118
6.6.3 System Operation 119
6.7 Summary 120

7 Evaluation 121
7.1 Introduction 121
7.2 VCUDSDC Framework 121
7.3 Visual Engine 123
7.4 Asset Lookup and Resource Monitor 123
7.5 Rule Engine 124
7.6 Scripting Engine 124
7.7 Object Behaviour Matching 125
7.8 Comparison with Existing Approaches 125
7.8.1 RUNES 125
7.8.2 Second Life 126
7.8.3 World of Warcraft 127
7.9 Framework Test and Evaluation 127
7.9.1 Evaluation of Framework to generate behaviour for objects. 128
7.9.2 SunSpot to Virtual Environment 130
7.10 Summary 131
8 Conclusion and Future Work 132
8.1 Introduction 132
8.2 Thesis Summary 132
8.3 Contribution to knowledge 134
8.3.1 3D Visualization in Ubiquitous Computing, Device and Services 134
8.3.2 Content Sharing across different Environments (Virtual and Physical) 135
8.3.3 Object Behaviour Matching 135
8.3.4 Interaction and Control in Ubiquitous Computing System 136
8.4 Future Work 136
8.4.1 Visual Engine 137
8.4.2 Rule-Engine and Ontologies 137
8.4.3 Security 138
8.4.4 Script Parser and Generator 138
8.5 Concluding Remarks 138
REFERENCES 141
APPENDIX A: VCHUDS USE CASE DIAGRAMS 151

APPENDIX B: TABLES 161
APPENDIX C: Log File for Computer A 162

LIST OF FIGURES

Figure 1.1 Visualising Ubiquitous Devices and Services 2
Figure 2.1 Texture 25
Figure 2.2 A Basic Production System [Proctor 2008] 29
Figure 2.3 Forward Chaining [Proctor 2008] 30
Figure 2.4 Backward Chaining [Proctor 2008] 31
Figure 2.5 Napster's centralised P2P model 36
Figure 2.6 : Gnutella pure P2P model [Science 2006] 37
Figure 4.1 Visualisation and Control of Ubiquitous Devices, Services and Digital Contents Framework 51
Figure 4.2 VCUDSDC Framework in Network 51
Figure 4.3 Component Sequence Diagram 53
Figure 4.4 Dynamically generating object behaviours 54
Figure 4.5 Start Virtual Environment 58
Figure 4.6 Load Objects 59
Figure 4.7 Load and Publish object Behaviour 60
Figure 4.8 Object Behaviour's Advertisement 61
Figure 4.9 Behaviour Advertisement Services 62
Figure 4.10 Generate Script using Rule Engine 63
Figure 4.11 Load Behaviours from Datasource 64
Figure 4.12 Find Behaviour 65
Figure 4.13 Script Validation 66
Figure 4.14 Visual Engine 67
Figure 4.15 UML-diagram of Visual Engine 68
Figure 4.16 Data Model 69
Figure 4.17 Assert Lookup/Resource Monitor 70
Figure 4.18 Rule Engine Class Diagram 71
Figure 4.19 Rule Process 72
Figure 4.20 Script Engine Class Diagram 74
Figure 4.21 Scripting Engine 75
Figure 4.22 IOPE Matching and dynamic generation of behaviour 77

Figure 4.23 Process Behaviour Request 78
Figure 4.24 Perform Abstract Match 79
Figure 4.25 Build Signature 81
Figure 5.1 Lamp 85
Figure 5.2 Smart Home Environment 86
Figure 5.3 TV 87
Figure 5.4 Mobile Phone 88
Figure 5.5 Frame to XML 89
Figure 5.6 Augmenting Frames 90
Figure 5.7 3D Scenery 90
Figure 6.1 VCUDSDC Framework 97
Figure 6.2 Room view in Blender 99
Figure 6.3 XML schema of the mesh file 100
Figure 6.4 Scence 3D model data for window 100
Figure 6.5 XML to JME 102
Figure 6.6 Media Player Object 103
Figure 6.7 Media Player in XML 104
Figure 6.8 Hierarchy of the Media Player 104
Figure 6.9 Scripting Engine 105
Figure 6.10 Scripting Error Box 106
Figure 6.11 Script Evaluation 106
Figure 6.12 LJMU Network appliances Lab 107
Figure 6.13 Rule Syntax 108
Figure 6.14 Rule to generate SunSPOT behaviour 109
Figure 6.15 Virtual Lab User Interface 111
Figure 6.16 Controls SunSPOT from Virtual Lab 113
Figure 6.17 Rendering information for a sunSpot Sensor 114
Figure 6.18 Rules used to create scripted behaviour 115
Figure 6.19 Mobile phone Script to call using Skype 117
Figure 6.20 Media Player 118
Figure 7.1 Time Taken (seconds) vs. Object Creation 129
Figure 7.2 Data transfer versus time. 131
Figure A1.8.1 Start Virtual Environment 151

Figure A.8.2 Connect To Network 152
Figure A.8.3 Process Behaviour Metadata 153
Figure A.8.4 Process Visual Metadata 154
Figure A.8.5 Create Behaviour Semantic Model 155
Figure A.8.6 Find Object Behaviour 156
Figure A.8.7 Evaluate Object Behaviour 157
Figure A.8.8 Process Behaviour Request 158
Figure A.8.9 Scripting Engine 159
Figure A.8.10 Visual Engine 160

LIST OF TABLES

Table 7.1 Behaviour generation and rendering for Lamp object introduced for the first time 128
Table 7.2 Numbers of times the Lamp was added to the Virtual World 129
Table 7.3 Average Time Taken for complete process for different objects 130
Table 8.1 Scenario Parameters 161
Table 8.2 Log File to add Lamp to Virtual World at run time 162
Table 8.3 Log file To add lamp to virtual world at run time 175
Table 8.4 Log file To add lamp to virtual world at run time 177
Table 8.5 sunSpot Object Log file 178

LIST OF ACRONYMS AND TERMS

JME	Java Monkey Engine
RGB	Red, Green, Blue
HSV	Hue, Saturation, Value
PAL	Phase Alternating Line
NTSC	National Television Standards Committee
SECAM	Sequential Colour with Memory
LWJGL	LightWeight Java Game Library
UML	Unified Modeling Language
DLL	Dynamic Link Library
JNI	Java Native Interface
AABB	Axis Aligned Bonding Box
OBB	Oriented Bounding Box
k-DOP	k-Discrete Oriented Polytop
XOR	Exclusive Or
JFC	Java Foundation Classes
HDV	High Definition Video
YUV	Luminance, Chrominance
API	Application Program Interface
JMF	Java Media Framework
Appliance	A device or instrument designed to perform a specific function
CC/PP	Composite Capabilities/Preferences Profile
CE	Consumer Electronics
DeCap	Device Capability matching service
DHWG	Digital Home Working Group – Home networking middleware
HTTP	Protocol used to transmit and receive files
IOPE	Inputs, Outputs, Preconditions and Effects
IP	Internet Protocol
JAR	Java Archive File – contains java resources to support the service
JVM	Java Virtual Machine
JXTA	Set of Peer-to-Peer Specifications
MAUT	Multi-Attribute Utility Theory
Middleware	Software that mediates between an application and a network
SUF	Service Utilisation Framework
Networked Appliance	A dedicated device with an processor and a network connection
Ontology	Formal specification for representing objects and relationships
OSGi	Open Services Gateway Initiative
OSI	Open System Interconnection
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
P2P	Peer-to-Peer
PC	Personal Computer
Pervasive	Manifested throughout; penetrating or affecting everything
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	RDF Query Language
Reasoner	Something that can find new facts from existing data
ROI	Regions of Interest
RPC	Remote Procedure Call
Service	A unit of work done by a service provider for a service consumer
Signature	A method name including its associated parameters

SSDP	Simple Service Discovery Protocol
Structured Service	Use third party software to register and advertise functions
SISM	Semantic Interoperability and Signature Matching
SOA	Service-Oriented Architecture
Vocabulary	All the words of a language
Ubiquitous	Being or seeming to be everywhere at the same time; omnipresent
UDP	User Datagram Protocol
Unstructured Service	Provides services independent of any kind of third party
UPnP	Universal Plug and Play – Home networking middleware
URI	Universal Resource Indicator
XML	Extensible Markup Language
WSDL	Web Service Description Language

Chapter 1

1 Introduction

1.1 Preamble

Many devices have developed rapidly to become multifunctional wonders. They now include functionality beyond their initial design. The best example is a mobile phone. They not only provide communication functions such as making a phone call and sending text messages, but can also work as a camera, MP3 (Moving Picture Experts Group Version 3) player and support web access. More and more devices are providing computing capabilities, including increased networking functions enabling them to interact with each other more easily. The multifunctional capabilities of devices have allowed for new and exciting application areas for networked appliances. For example, using the network, devices can be controlled from anywhere in the world at anytime.

Equipped with multiple networking capabilities, increased computational power, and higher data throughput, many devices have become ideal platforms on which multimedia content can be distributed and accessed ubiquitously. Consequently, this has allowed the source of multimedia to change beyond traditional distribution models. For example, sensors over the last several years have been the focus of much research and are now starting to form part of larger networks designed to monitor environmental conditions such as temperature, vibration and pollutants, often used as input for 3D visualisations.

Solutions may manifest themselves from the bespoke deployment of ubiquitous devices and services or by exploiting existing networked consumer appliances. However, regardless of how such devices and services embed themselves or how invisible technology, becomes the challenge will be to invent and develop a new platform capable of making devices and services accessible. Drawing from advances made in game technology techniques could be utilised to form a bridge between real

and virtual worlds as illustrated in Figure 1.1. This would allow the constraints to be mitigated, because virtually, physical devices can be manipulated in new and novel ways, i.e. increasing their size to make them more visible; or by combining them with other devices to provide a better means of interaction. For example, a digital magnifying glass would allow us to see what invisible technology looks like and how we may use it. We could even locate its position within the home. Given that there is widespread adoption of networked appliances, all providing an increased number of functions, the home will contain many invisible services. New platforms are required to exploit these services to allow users to include them in any number of different applications. Successful platforms will also provide greater visibility to those services, from outside the home to other interested parties, for example, to monitor a patient's health.

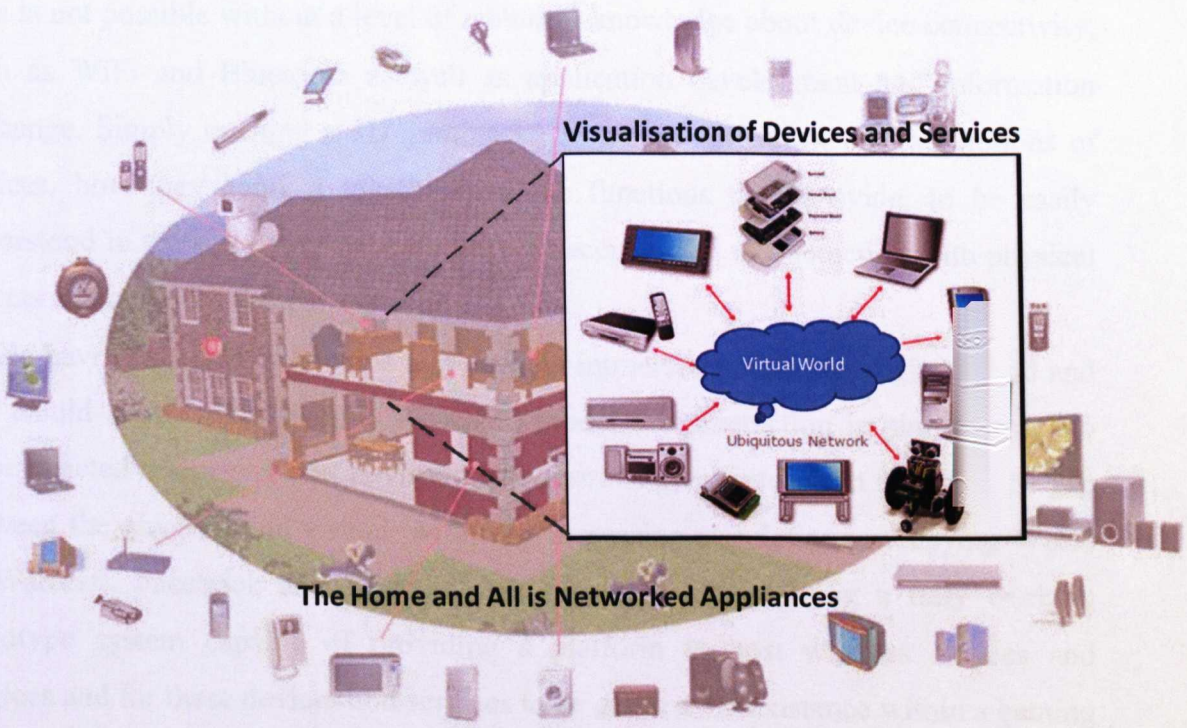


Figure 1.1 Visualising Ubiquitous Devices and Services

In the remainder of this chapter we provide an overview of the challenges that need to be addressed and discuss their importance. A brief introduction is provided about the research fields considered within this thesis where all concepts relating to its construction are clearly defined. This includes networked appliances, home networking, 3D virtual environments, Object Behaviour Matching, dynamic

behaviour composition and ubiquitous computing. Current techniques and research practices are described and their associated strengths and weaknesses are highlighted. Finally, we conclude this chapter by defining the scope of this thesis, the key requirements that this thesis addresses, the novel contributions we have made and an overview of the remaining chapters.

The motivation for this research project is to bring together networked appliances and mobile computing devices, including devices that are not necessarily easy to see such as internal or external body sensors. Utilising wireless communications and an intelligent service-oriented middleware platform it is easy to find and use functionality provided by all our devices and this makes them easy to group, manage and control in a much more easy and natural way through advances in entertainment technologies such as 3D game and physics engines. This allows us to move a conceptual magnifying glass over our environments and magnify what is available. This is not possible without a level of technical knowledge about device connectivity, such as WiFi and Bluetooth as well as application development and information exchange. Simply opening a 3D gaming environment will allow abstract notions of devices, how they connect together and the functions they provide, to be easily understood in the same way we have become accustomed to interaction with physical objects and our environment.

We have undoubtedly become increasingly immersed within our digital world and this would suggest that there is a need to model this interaction in similar ways to those enacted in our physical environment. There is great interest in bridging the gap between the physical and virtual, with ever increasing populations occupying World of Warcraft, Facebook and Youtube. We take that further using a fully working prototype system capable of providing a platform to host wireless devices and services and for these devices and services to be given a 3D existence within a gaming world where they can be viewed and acted upon in ways that extend what is currently available in the 3D Web. The remainder of this chapter introduces the key technology components that when combined provide a new and novel platform to visualise and control ubiquitous devices, services and digital content.

1.2 Ubiquitous Devices

As we move towards the Ubiquitous Computing vision, our homes will adapt to embrace the benefits this will bring. Whether we will be completely aware of this transition is unclear. For example, in the UK it is reported that we are caught on CCTV (Closed-Circuit Television) cameras on average 300 times a day, which many of us are unaware of [Welsh 2009]. Like CCTV, technologies will weave themselves into our home networks to influence all aspects of our home life, without us necessarily being aware that it is happening. This evolutionary path has been slowly pushed over the past five years through the wide deployment of broadband and wireless solutions and this has allowed us to use the home in new and emerging ways. For example, through cellular networks and machine-to-machine technologies we now have automated utility usage and billing services. We can seamlessly move digital content around the home with wireless devices providing us with ubiquitous access to our content. And we can interact with physical artefacts, i.e. using technologies such as Surface Computing [Steve 2007].

Coupled with sensing technologies, such as RFID (Radio-Frequency Identification), infrared and new gaming controllers currently being released by Sony and Microsoft, these devices will play a greater part in embedded home services. For example, we may see the deployment of medical sensors in the home, in what is being termed The Medical Home that could be directly linked to body sensors for the purpose of collecting data, i.e. physiological and motion data. This could have many benefits to the user. Firstly, biofeedback data could be used to control devices, such as those that influence mood, entertainment, or game play. Secondly, they could help support next generation primary care services from within the home to monitor, diagnose and treat any number of ailments such as, physical and cognitive decline, including debilitating conditions, such as depression or Alzheimer's.

Given that there is widespread adoption of networked appliances, all providing an increased number of functions, the home will contain many invisible services. We developed a framework that allows us to exploit these services to allow users to include them in any number of different applications. Using a virtual environment will provide greater visibility to those services, not only within the home but from outside the home by other interested parties, For example, to monitor a patient's health.

1.3 Networked Appliances and Home Networking

Networked appliances [Nakazawa 2000] can be described as consumer equipment with a dedicated function which has a network process that allows the device to be connected to a network [Merabti 2008a]. Networked devices have revolutionised the way we access and disseminate information and changed the way we communicate with each other. Integration between different domains, such as the Internet, broadcast and mobile, has now been made possible using standards such as DLNA (Digital Living Network Alliance) [DLNA 2004a; DLNA 2004b]. Furthermore Set-top box technologies have increased in popularity and standards such as the OSGI (Open Services Gateway Initiative) [OSGi Alliance 2005] have helped enhance these solutions to allow services, such as audio and video, to be deployed on network appliances connected to home networks. Keeping with the view that many more devices will form part of the Internet, the RUNES (Reconfigurable Ubiquitous Networked Embedded Systems) [Koumpis 2005] project has provided a framework that allows any device to form part of the Internet irrespective of its capabilities. This is seen as an important step where advances in sensor networking technologies have emerged to be new sources for multimedia data.

Despite the benefits such technologies bring many are still proprietary in nature. Adapting and controlling how services are created, deployed and managed is somewhat restrictive. This has resulted in more flexible approaches, for example those based on peer-to-peer technologies, where devices can form part of larger networks in a much more *ad hoc* manner. AMIDEN [Minoh 2001] represents a platform where functions provided by devices can be dispersed within the network as independent discoverable services. A standard that helps facilitate the exposure of device functionality in a more service-oriented manner is JXTA (Juxtapose) [Gong 2001]; which is a set of protocols designed to create a balance by providing a hybrid system that uses loosely consistent distributed hash tables. The SIP (Session Initiation Protocol) is responsible for the setup and release of connections between two provided end points [Tam 2002]. SIP is widely used in VoIP (Voice over Internet Protocol), to setup, terminate and configure call and data transfer. SIP usually runs on TCP/IP (Transmission Control Protocol/Internet Protocol) but can be run on UDP (User Datagram Protocol), ATM (Asynchronous Transfer Mode Protocol) and a frame

relay running over TCP is advantageous because it provides inexpensive widespread connectivity, directory services, naming services and a widely known development environment. MULEs (Mobile Ubiquitous LAN Extensions) are mobile entities [Shah 2003], that provide short-range wireless communications and can transfer and receive data from nearby sensors or access points. When a sensor is in close range, MULEs stores sensor data temporary in a buffer and transfers data to a wired access point when in proximity [Akyildiz 2002].

There are many solutions available for network appliances and home networking from self-adaptive middleware frameworks that enable heterogeneous networks, devices and services to be seamlessly interconnected [Merabti 2008a] to bespoke solutions. However, in home networking there are hidden services or small devices which cannot be operated physically. The goal of our research is to visualize networked appliances and these hidden services and small devices in 3D virtual environments to support possibilities for device interactions and operation and to allow the physical constraints associated with real objects to be removed [Shaheed 2009].

1.4 Virtual Environment

A 3D virtual environment is a computer simulated environment where a user interacts via avatars in Games such as Halo, World of Warcraft or virtual social networks like Second Life. The virtual environment is used to simulate reality where users can control the virtual environment [Simin 2009]. Traditionally virtual environments have been controlled using keyboards or joysticks but later advances in sensor technology have made it possible to use other input devices such as motion capture, hearing sensor, touch sensor etc. Flight simulator [George 2008] is an example of a virtual reality environment where pilots are trained using computer programs before they use a real airplane. Other virtual environments have been developed for different application such as military exercise, 3D medicine imaging and entertainment.

The aim of virtual reality technology is to bridge the gap between the virtual and physical and to explore man-machine interfaces from visual perception and different

sensory channels such as vision, hearing, force, touch and smell. Advances in sensor technologies will make the experience more immersive [Fergus 2007].

The ability to change the internal look and behaviour of virtual environments has extended beyond those mentioned above where we now see devices embedded in the real-world; these devices are beginning to affect virtual environments and the artefacts they contain. For example, the study by Kwon et al., [Kwon 2007] shows how virtual objects can be visualised using information collected in wireless sensor networks. This provides virtual worlds with the ability to understand and estimate more easily the state of the real world measured through these sensor networks.

Many 3D virtual environments exists where people can create their own content, buy and sell virtual products, and connect physical devices with virtual worlds. However little has been done on sharing contents across different environments, for example moving an object between Second Life and World of Warcraft. The challenge is to design a framework to facilitate sharing of objects across different virtual environments and allow such environments to create behaviour for object. More specifically, we have developed a framework that builds on the many advances discussed above. The benefits of each are exploited to allow user generated content and the seamless inclusion of devices and the functions they provide in the real world to be seamlessly moved, shared, and used within and across different real and virtual heterogeneous environments [Merabti 2008b].

1.5 Incorporating Rule Based System and Dynamic Scripting

A Rule-Based system processes and stores knowledge to interpret information in a useful way. Using Rule-Based systems the visualisation and operational capabilities of objects can be manipulated externally to the applications in which they are shared and used. New and novel platforms will require intelligent modules capable of understanding and making interpretations about visualisations and the behaviours objects have to provide mechanisms to plug services into the network that can act as helper functions [Merabti 2008a]. For example, object visualisations serialised in COLLADA [Erwin 2007] [Remi 2006] (COLLAborative Design Activity) from the source environment could be transcoded into multimedia authoring tool such as 3D

Studio Max if used by the destination environment using the framework [Merabti 2008a]. Behaviours are slightly different in that it is the target environment and its specifics that are best suited to dynamically creating such behaviours. Initially objects that are projected into target environments describe their functions semantically. One possible way of doing this could be to utilise advances made within the Semantic Web [Berners-Lee 2001] and the tools they use such as OWL-S [Martin 2005]. These semantic descriptions can be used as input into services equipped with rule engine capabilities. When semantically matched rules fire [Merabti 2008a] dynamic scripted behaviours are generated and serialised using script languages such as Ruby, Python, Lua or JavaScript. These scripted behaviours are then projected alongside the visualisation data into the target environment.

There are many options for incorporating scripting into games and networked virtual environment applications, by utilizing an existing solution, or developing a custom bespoke tool. The bespoke tools can be very powerful, and provide the developer with the ultimate control over the solution, but also involve an extremely large amount of development to create them from scratch, putting this method in conflict with the productivity gains made from utilizing scripting. When adopting a generic scripting language approach, all four of the languages mentioned provide an adequate range of language features, high-level bindings to other languages, built in support functionality, and portability. The choice will be application dependent e.g. using Lua if speed and memory footprints are crucial, Python if existing functionality must be utilized, and Ruby for maintaining an Object-Oriented approach.

There are many 3D virtual environments, both in games and social networking, which support scripting languages, Rule-Based systems and different virtual environments support scripting with some having their own scripting language support for example Second life. However, the challenge is to allow users to choose their preferred scripting language and for environments to support that language. We have developed a system where it will be easy to select a popular scripting language and write code in that language using a rule based system and incorporating semantic matching techniques to find the appropriate behaviours of an object [Shaheed 2008].

1.6 Scope of the Research

The aim of this thesis is to develop a new framework that allows us to discover, visualise and control devices and services in Ubiquitous Computing systems. By equipping devices with visualisation and behavioural data they are able to project themselves in virtual environment. When discovered, this data is stored and mapped to objects (visualisation data and semantic descriptions of data) in a virtual container. This information is pulled into the framework and used to create interpreted visualisation information and dynamically scripted behaviour from the semantic descriptions of functionality the object provides.

The framework can be easily expanded to accommodate any additional requirements to be plugged in as and when they are needed. This thesis focuses on discovering new services, visualisation, constructing behaviour on the fly and sharing objects in heterogeneous environments (both virtually and physically).

Using this framework several key requirements are addressed within this thesis, which encompass advances made in the areas of networked appliances, dynamic generation of object behaviour, visualisation, semantic matching and ubiquitous devices. The framework does not consider the aforementioned disciplines in isolation, but rather proposes a service architecture demonstrating how they can be combined and extended to create a new type of framework capable of discovering devices, generating their behaviour on the fly and visualizing them in a virtual environment.

1.7 Project Requirements

This section presents five main requirements used to design and implement the new framework and to realise the challenges described in this chapter.

- Bridging the gap between physical and virtual worlds will allow us to remove the physical constraints associated with real-world objects that will be beneficial in two ways. First, improving possibilities for device interactions and improving the functionality that is available within virtual worlds. Second, it will also allow the physical constraints associated with real objects to be removed, enabling them to benefit from the freedom offered by virtual environments. This will require new algorithms that link devices and virtual

artefacts as well as moving them between heterogeneous environments. There is a need to create artefacts that are modular and descriptive in terms of their appearance and their behaviours. Different environments have different rules governing how they contain artefacts and behaviours, consequently services will be required to tailor artefacts in conformance with environmental laws. This means that services need to transcode 3D model data and dynamically support behaviour development using a combination of on-the-fly scripting and semantic reasoning using a logic system such as a rule engine [Maciol 2008]. This will improve interactivity and the way the platform allows manipulation and control of content, to the extent that new user-generated content can be constructed purely through the combining and manipulating of 3D multimedia streams using networked appliances.

- The platform should effectively generate dynamic behaviour and allow us to map different object behaviours using Ontologies and semantic descriptions to describe content visualisations and behavioural data (the interaction between objects across different virtual environments). Behaviour ontologies allow objects to semantically describe the functions they provide and for these to be embedded within object advertisements. Using the same ontologies requests for objects or a description of required behaviour can be described.

- Object Descriptions and Object requests must be based on environment-processable semantics if we are to successfully determine what behaviours are relevant and which are not. This brings with it additional challenges; the vocabularies used by different environments will be different and the structure of the concepts themselves will vary. Therefore, we need mechanisms that allow environments to dynamically create a semantic interoperability bridge between terms that are syntactically different but semantically equivalent. This requirement allows environments to discover other objects and behaviours within their environments and dynamically learn the different terminology they use.

- With the advent of social networking and more importantly virtual environments in which such social activities take place it has become important to understand how interoperability between such environments can

be realised. Users need to have the ability to move seamlessly between heterogeneous environments where content from such environments can be shared. Typically, this is not the case where proprietary platforms host virtual environments with limited or no openness to include others. Given these limitations, we aim to provide an open platform on which any virtual environment can interoperate with another and share, use, and create content in any of them. For example, content owned by users in Second Life can be shared with a World of Warcraft user.

- Communication underpins virtual environments and this could possibly be the beginning of the 3D Internet. This gives virtual environments a greater reach and one that will more and more include devices from within the real-world. For example, sensor networks will be used to obtain environmental data and provide a direct effect on content in virtual environments. Users will be able to utilise the devices they have from within virtual environments and vice versa. Consequently, there is a need to develop protocols that blur the gap between the two. Building virtual worlds on top of a networked appliances platform that forms part of the Internet, is unique, both in terms of its design, and the level of user control that the resulting platform will provide.

1.8 Novel Contributions to Knowledge

This thesis describes a new framework we have developed that uses advances in gaming technology to address the above project requirements by discovering ubiquitous computing devices, visually representing what they look like and mapping the functionality, they provide. Our framework provides services and protocols that discover and interconnect objects within the network; dynamically generate and compose functionality provided by objects using semantic matching; select objects based on the capabilities they support; and allow environments to generate behaviour dynamically. Each of the novel contributions we have made are discussed in turn in the following subsections.

1.8.1 3D Visualisation in Ubiquitous Computing, Device and Services

Currently applications are developed and deployed as one-off solutions – any application changes thereafter appear in subsequent releases. Although such

applications provide considerable benefits, it is becoming increasingly apparent that these solutions are inflexible. Alternative mechanisms are needed that allow application functionality to be embedded within the environment as network services. This will allow new frameworks to utilise these services to create complex business processes more quickly. We have developed such a framework that allows the operational functions provided by devices to be dispersed within the network as services that can be combined to create high-level applications [Fergus 2003; Mingkhwan 2004; Mingkhwan 2005; Fergus 2005] and these services can be provided and exploited by any type of network devices. Each contribution we have made is listed below:

- We have designed a framework that allows application functionality to be embedded within the environment as network services. The operational functions provided by devices to be dispersed within the network as services that can be combined to create high-level applications and these services can be provided and exploited by any type of network devices [Shaheed 2010].
- Our Framework allows devices to be visualised in virtual environments in order to enhance the device functionality and hidden services within the network in a unique and novel way to give greater control over services and devices within networks.

1.8.2 Content Sharing Across Virtual and Physical Environments

Content sharing over networked devices, beyond simple file sharing is becoming a reality. Furthermore, many devices are forming closer relationships with different virtual worlds, such as World of Warcraft and Second Life. In one sense the gap between the two is becoming increasingly narrow. Consequently, this opens up many new avenues for content sharing, not only between devices but also between sophisticated virtual worlds. This will open up new opportunities where third-party content providers and users alike will be able to create and share content over these new platforms. This provides obvious benefits. First, the freedom this offers allows us to very easily move between real and virtual environments where the content and services we use are always at our disposal. Second, utilising the benefits of being

connected will allow us to manage our content and services independently of where they reside, whether this is on the device itself or remotely via its associated avatar. Lastly, and perhaps less obviously the platform will significantly influence the applications underpinned by digital content where solutions not yet envisioned will emerge, such as dynamic game development, immersive and interactive 3D multimedia, and on-the-fly scene analysis and manipulation of object behaviour. We aim to provide a basis on which this vision can be realised where mechanisms have been developed that facilitate the sharing of virtual world objects across different virtual environments. The approach has been successfully developed and tested using a working prototype that allows digital content, such as 3D assets to be shared; physical devices, such as mobile phones, to be connected to avatars in a virtual environment, and content to be shared. Each novel contribution is listed below:

- We have developed a component to plug into any environment (both physical and virtual) which eases the sharing of objects and the generation of behaviour dynamically in the target environment. It is a difficult task to generate behaviour of objects dynamically in target environments, which are not known a priori because this will possibly bring an infinite number of behaviours. The environment (both real and virtual) generates behaviours on-demand and determines how a particular behaviour is used. In heterogeneous environments, generating dynamic behaviours using pre-determined interfaces or implementing specific proxies are not possible because we have no control over objects joining the environment. We dynamically generate behaviours and 'intelligently' process the signatures they provide to allow the environment to understand how the object operates and how they can be integrated within different environments [Shaheed 2007].
- Current implementations describe object behaviours using attribute-value pairs. This means that successful matches are only found if the behaviour request exactly matches the object description. If the two differ syntactically, but are equivalent semantically, current approaches fail to find a match. This is inflexible and excludes a large number of behaviours because of syntactic differences. In our framework we provide mechanisms that serialise behaviour descriptions using high-level semantics that provide rich conceptual information about the individual functions devices provide [Fergus 2003a;

Fergus 2003b]. Even if behaviour requests and behaviour descriptions are syntactically distinct, but semantically equivalent, our framework can find a match.

- It is difficult to get different content makers to create and use a single standard for the terminology used to describe object behaviour. Consequently our framework uses high-level semantics to resolve the inherent ambiguities between object behaviour requests and behaviour descriptions [Fergus 2003a].

1.8.3 Object Behaviour Matching

Content Sharing between devices and their usage scenarios in the real world does not entirely map onto shared content and devices and their usage scenarios in the virtual world. We have a level of intelligence that makes this process easy, which is not shared with computers. This is a problem well documented within Artificial Intelligence (AI), however, aspects from AI can be utilised. We borrow from many of the successes seen in the Semantic Web [Merabti 2008a], more specifically the use of semantic descriptions to describe content visualisations and behaviours (the interaction between objects across different virtual environments).

Behaviour ontologies allow objects to be semantically annotated, which can be embedded within object advertisements. The same ontologies can also be used to describe object requests. We borrow from the Networked Appliance Utilization framework [Merabti 2008a], and use object descriptions to describe behaviour ontologies; the Behaviour Ontology; Behaviour Profile; and the Behaviour Process Model [Shaheed 2008]. Behaviour ontologies allow objects to be described at an abstract level in terms of IOPEs (Inputs, Outputs, Preconditions, and Effects). The IOPEs form explicit relationships between the different ontologies, which are in turn mapped into signatures (method names, parameters and return data including type information). In this way the Behaviour ontologies and the behaviour interface provide a mechanism to link semantic descriptions for example WSDL (Web Service Definition Language) to possible implementation solutions to describe how behaviours are generated. This process helps independent behaviours offered by objects to be dynamically created or discovered, and executed with little or no human intervention.

- Our novel contribution resides in the fact that content is not something that resides in a location in which it was created but something that is invisibly connected to content owners. Our framework allows content to be metaphorically dragged between content services and devices through highly interconnected networks dependent on what device we use or which environment we inhabit [Shaheed 2008].

- Our framework combines semantic annotations (behaviour ontologies) with rules to create dynamic scripting on the fly and to project them into heterogeneous virtual environments. We have moved a step further than other approaches, such as modding or world construction as in Second Life, to creating dynamic environments containing different artefacts with corresponding appearances and behaviours. This gives the user greater power to manage and use their content and functions independently of the constraints currently found in different virtual environments.

1.8.4 Interaction and Control in Ubiquitous Computing System

Ubiquitous Computing, according to Weiser's original paper is the aspiration to weave technology throughout the environments we inhabit and into every aspect of our daily lives [Weiser 1991]. Using this definition, many research strands now investigate the use of small devices within the environment that are capable of transferring and receiving information. Applications are built around sensors that dynamically configure themselves dependent on the user's context. Whilst we are still some way from a complete implementation of this vision aspects have already been developed. For example, solutions based on Bluetooth have been proposed for proactive information delivery [Ballance 2008]. Based on its success other protocols have been proposed, for example ZigBee [Geer 2005].

This evolutionary path has been slowly pushed over the past five years through the wide deployment of broadband and wireless solutions and this has allowed us to use the home in new and emerging ways. For example, through cellular networks and machine-to-machine technologies we now have automated utility usage and billing

services. We can seamlessly move digital content around the home wirelessly providing us with greater ubiquitous access to our content. We can interact with physical artefacts, i.e. using technologies such as Surface Computing. We have made the following contribution to this area:

- We have developed a novel framework that describes how we can discover, visualise and control devices in Ubiquitous Computing systems. By equipping devices with visualisation and behavioural data they are able to describe themselves. When discovered this data is stored and mapped to plug-in data (visualisation data and semantic descriptions of data) in the virtual container. This information is pulled into our framework and used to create interpreted visualisation information and dynamic scripted behaviour from the semantic descriptions of functionality the object provides. We exploit the concept of rules to understand these semantics and govern how behaviours are constructed in conformance with the scripting language supported by the destination environment and the behaviours it can support [Shaheed 2010].

1.9 Thesis Structure

In this chapter, we provided a general idea of the problem domain, namely the inefficiencies related with current environments (both physical and virtual). It highlights that some work has been carried out within ad hoc home networking environments, and mechanisms for enabling objects to move around and connect physical devices to enhance functionality in virtual environments. This chapter argues that content needs to be shared across different environments (both physical and virtual) and functionality should be generated on the fly. In doing so, the challenges are presented, which include - behaviour generation on the fly, behaviour discovery, rules, program scripting and ubiquitous computing. This chapter also introduced a framework we have successfully developed that addresses these challenges. Finally, the chapter is concluded by defining the scope of the research project, the novel contributions made and an outline of the thesis structure.

In chapter 2, we begin by presenting the background within the field of virtual environments and social networking. This discussion defines the key concepts used within this thesis. This also discusses different Rule-Based Systems and scripting languages, Peer-to-Peer networking and discusses how scripting is essential to design a mobile environment where new objects and their functionality can be created on the fly. The Rule-Based system provides a mechanism to apply business logic and modify the logic at run time.

In Chapter 3, we discuss work related to ubiquitous computing and networked appliances. In this chapter, we discuss in detail different networked appliances approaches related to home networking and explain current middleware solutions that intend to interconnect devices within home environments.

In Chapter 4, the design methodology used to describe the system and its deliverables is discussed. As part of a solution description, the formal design of the Visualisation and Control of Ubiquitous Devices, Services and Digital Contents Framework is presented (using UML), detailing the different components of the framework and the overall communications between them.

In chapter 5, a Smart Home Environment case study is presented which describes how the new framework implementation can be used to facilitate content sharing and to discover devices and hidden services in virtual environments. This case study also describes how devices can be connected with virtual environments, how its behaviours are generated on the fly, and how we can control physical devices from virtual environments.

Chapter 6 discusses the implementation details for the new framework - the different toolset used and their impact on the overall system. This chapter also includes the specification framework and explanations of which tools where used to address the key requirements within the framework.

Chapter 7 discusses the evaluation of our framework and its overall performance. Each component is evaluated with its associated functions. This chapter also compares our approach with other similar solutions.

Finally, Chapter 8 summarises the main contributions made in this research and discusses possible future work.

Chapter 2

2 Virtual Environments, Rules, Dynamic Scripting and Peer-to-Peer Networking

2.1 Introduction

This chapter provides a background on the state of the art work carried out in the main areas relevant to this thesis, which includes Gaming, Entertainment, Social networking, Rule-Based Systems, Dynamic Scripting, and P2P networking. In the following sections we will discuss computer games, social networking, an, overview of Rule-Based system and Scripting Engines, and P2P.

2.2 Computer Gaming, Entertainment and Social networking

Entertainment can be classified in different forms. A Computer game is an interactive entertainment which involves interaction with a user interface to generate visual feedback on a video display [Barr 2007]. In a movie, a scriptwriter writes a story and executes that story but audiences cannot interact or influence the characters – they simply watch it unlike in games where users interact and change the outcome of the story depending on the skill of that user.

2.2.1 Computer Games

The history of computer games dates back to the 1960s where it was typically based on text instead of graphics like we see today. The first computer game was developed by Martin Graetz and Alan Kotok with an MIT (Massachusetts Institute of Technology) employee Steve Russell on a PDP-1 computer in 1961 called Spacewar [Levy 1984].

Using the Internet, people play video games in their homes, alone or with friends. The industry has moved from designing computer games for LAN (Local Area Network) to massive distributed games where people can play with each via the Internet. For example, the PlayStation 3 and Microsoft Xbox 360 consoles now allow people to play games over the Internet [Messinger 2009].

The improvement of console technologies has changed the way people are entertained [Messinger 2009], the Internet and high speed broadband have rapidly developed the massively multiplayer gaming market with companies collectively pulling in $18.85 billion dollars in global sales in 2007 ($9.5B in game sales and $9.35B in console sales) .

Single player games are referred to as games where one player is involved against a machine without being connected to another machine. LAN games were first introduced in the early 90s for multiple players over LAN or null modems.

Massively multiplayer online games already attract huge numbers of players and are expected to become increasingly popular where they are already forming the basis for next-generation gaming. Utilising Internet communications, games have blurred virtual and physical worlds and converged with social networks [Seay 2004]. This has changed how users view and play games. Many games such as Planetside [Planetside 2006], Star Wars Galaxies [StarWars 2006], The Sims Online [Sims Online 2006] and EVE Online [EVE Online 2006], are dependent on network communications.

Although multiplayer gaming clearly provides significant benefits over single-player games with networking, its architecture enforces a number of limitations. Most notably, game play and enhancements must be carefully controlled through centralised gaming servers. This results in bottlenecks, central points of failure, and the inability to appropriately react to real-time changes in large virtual worlds. Gamers are tied to games through proprietary software and hardware installations. User interactions do not affect strategic developments and games do not support self-management capabilities to extend functionality beyond those they have been pre-

programmed with but there is some games which allow certain level of control over contents such as but they don't allow you to exchange contents with other games

This has lead to shifts within the gaming industry, where increasing access to game engines, software development kits and level editors has allowed games to be changed more easily. This phenomenon – known as modding – marginally alleviates some of the limitations discussed above [Sotamma 2005; Kucklich 2005; El-Nasr 2006]. Although modding provides a means of adapting and evolving games, it is restricted to more technically savvy users, such as software developers, rather than people who simply just play games. Furthermore, mods are tied to specific games. For example, a mod developed for the unreal engine will be incompatible with the *quake engine* [Quake 2010]. Some researchers suggest that distributed technologies in conjunction with middleware may relieve many of these difficulties, however it is generally accepted that more research is required to establish a suitable architecture [Hsiao 2005].

Modding is an activity that runs alongside mainstream games development, with developers providing modding tools as a way to attract customers. In essence modding is seen as a business strategy. Although not explicitly stated, incentives to mod games are used as a means of generating free development for publishers. For example through the use of modding competitions that act as a means of screening game enhancements in order to include them in future releases. In most cases this is an unpaid source of labour and gaming organisations carefully control how it is executed [Sotamma 2005]. Through competitions and gaming subscriptions for massively multiplayer online games, the industry has a healthy flow of mod software. In support of this several game companies adopt the principle of modding as a key strategy, where only a base solution is initially provided. Any enhancement to the game thereafter is dependent on user modifications. One example of this is BioWare's Neverwinter Nights, which is heavily reliant on gamer-created content [Sotamma 2003]. Successful mods have been incorporated into subsequent releases. Another example is Counterstrike, which is a modification for team play of Valve Software's Half-Life [Sotamma 2003]. In this case modding can be seen as an important and

welcome source of innovation where commercial risks are not taken by the gaming industry, but rely on the goodwill of the modders [Kucklich 2005].

The game World of Warcraft, became the fastest selling PC game in North America in 2004-2005 and in 2008 was reported to have ten million subscribers worldwide [Ducheneaut 2006]. World of Warcraft is one of the most popular massively multiplayer games. With the emergence of MMORPGs (Massively Multiplayer Online Role-Playing Games) such as World of Warcraft, players have turned to online writing communities to discuss game activities, strategies, and problems [Lee 2007].

The World of Warcraft (and MMORPGs in general) is not just popular for very high quality graphical appearance of the world and the conventional demand of being a monster killing game, but to be credited to the player to player teamwork and interaction mechanisms implemented in the game. The players work hard for each other to gain better items due to the architecture design of World of Warcraft where it is too hard for one player to play the game than playing as a team. If a player plays as a group then they can achieve better results[Kurniawan 2008].

2.2.2 Social Networking

"A social network is a configuration of people connected to one another through interpersonal means, such as friendship, common interests, or ideas. Social networking was not created in the age of the Internet; it existed long before. Social networks exist because humans are societal and require relationships with other humans in order to survive" [Coyle 2008]. Advances in computer technology has made it easy for people to connect with each other over long distances where they can share ideas.

Early examples of social networking saw people posting messages to each other using services like BBS and CompuServe, this was restricted to communications using text messages over short distances due to long distance charges. However, in the early 90s the Internet revolution changed everything and by the mid 90s it became cheaper to connect to each other over longer distances.

Social Networking websites were developed to allow people to create profiles, share photos, start blogging, and keep track of people they know through web applications like Class Mates [ClassMate 2010] but it did not allow users to make new friends. In 1997 Six Degrees of Separation allowed people to make new friends and exchange photos. In 2000 new Social Networking websites were developed with better functionality, such as Facebook.com and Myspace.com which allow people to socialize..

3D virtual worlds are three dimensional simulated environments where people interact using an avatar. Gaming and virtual worlds are beginning to play a critical role in Social Networking, business, education, technological sciences, and our society as a whole [Messinger 2009].

Consequently, there are many business opportunities and challenges involved in these environments, where millions of people all participate in the network to play online games. Papagiannidis *et al.* [Papagiannidis 2007] particularly discuss the principles and policies related to the social implications of Second Life [Life 2010] which raises significant research questions. One of the important questions, for example, is the payment issue either to the avatar or customer, while another issue is taxing people who are earning money in the virtual business.

Communication between avatars is most often conducted in a written format, either through chat or through instant messaging, although a voice-chat option was introduced in August 2007. In Second Life user can move from one location to another, using different mean such as walk, fly, teleport, or ride vehicles such as cars, submarines, or hot-air balloons.

2.3 3D Virtual Environment Technologies

In this section, we will discuss the technologies and tools that are used in the development of 3D virtual environments. 3D virtual environments have been used in our research as our focal point to visualise devices and services in home networks and to control them from within the virtual world as discussed in detail in chapter 1.

2.3.1 3D Modelling

In computer graphics, 3D modelling is the process of describing a three-dimensional object by its mathematical characteristics in a mesh representation using specialized 3D software [Clinton 2008] Modelling 3D objects takes much time, needs patience, is very complex and it needs some skills in art [Son 2009]. Below are listed some popular 3D modelling software applications.

- 3D Studio MAX
- Cinema 4D
- Zbrush
- Quidam
- Maya
- 3D Canvas
- MilkShape 3D
- Anim8tor
- YASRT
- Blender

2.3.2 Positioning in Space

The basic structure of the modelled virtual environment is a huge cube. Its centre is the point of origin of the Euclidean space [Franke 1995]. All the objects of the virtual environment are positioned around this centre point. The dimension and scale of the individual objects are very important. Like in the real world, it is not possible that a 15 inch flat screen is bigger than the table it stands on or a chair is smaller than a cell phone. We have to consider the optimal dimensions of the entire model. For example, tools such as Blender make it possible to zoom in and out to see how big or small the individual object is in relation to its environment.

2.3.3 Textures and Colours

In order to give every object a realistic look in computer graphics it is common to use textures. Using textures it is possible to let a table, consisting of multiple cylinders, look like a wooden table with all its grain. For example, we only need to put a picture with the grain onto the surface of the virtual object. This process is called texture mapping [Eberly 2007]. The advantage of textures in comparison to models is lower complexity, which means less computing time and less memory space usage. In Figure 2.1 the window is occupied with a texture (image taken from a camera connected to the network), so that the effect of an outside world is given. In the lower left corner a preview is shown of the picture which can be seen after rendering on the pink marked window in the middle of the Figure 2.1.

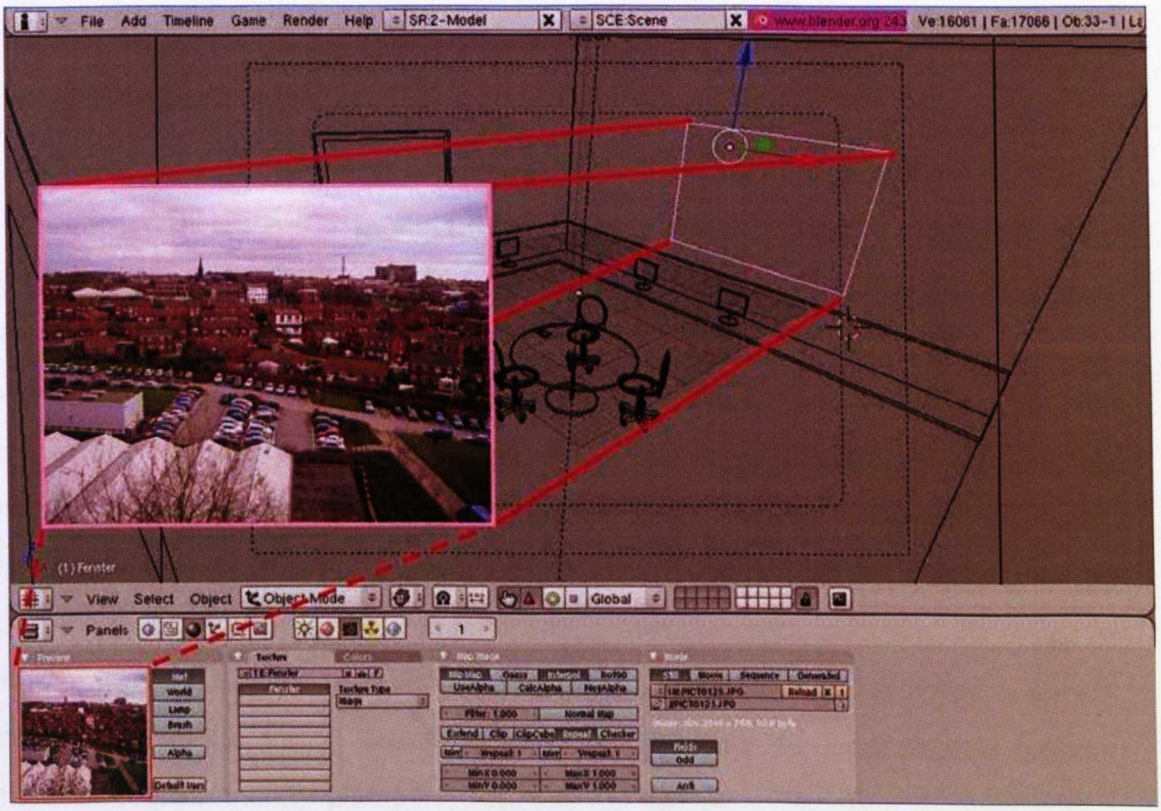


Figure 2.1 Texture

The most common colour model is the additive RGB model. The primary colours red, green and blue are added together to white. They are combined in various ways to reproduce other colours. A colour is described by three values, the red, green and blue fractions. In pictures with 8 bit per channel, every colour fraction can vary between 0 and 255.

The HSV model, also known as HSB (hue, saturation, brightness), defines a colour space in terms of three single components:

- Hue, the colour type (red, blue, yellow, etc.)
- Saturation, the intensity of the colour
- Value, the brightness of the colour

The YUV model defines a colour space in terms of one luma and two chrominance components. Y stands for the luma component (brightness), U and V are the chrominance (colour) components. It is used in the Phase Alternating Line (PAL), National Television Standards Committee (NTSC), and Sequential Colour with Memory (SECAM) composite colour video standards.

2.3.4 3D Gaming Engine

The Task of a 3D engine is to manage all 3D objects and scenes in the virtual environment [Ahearn 2007]. In addition, the engine interprets what is displayed on the screen and how it is drawn. There is a higher and lower level for the respective decision. The higher level is assumed by the application itself with the help of the scene graph. This happens at software level. The lower level is assumed by the renderer and occurs at the hardware level, especially on graphics processing units. To render a two-dimensional picture on the monitor screen from the information of three-dimensional objects in a 3D environment, a virtual camera, textures and lights are processed by the rendering pipeline. The final process to create an actual 2D image or animation from the scene is called rendering [Ma 2003] and 3D rendering is an art-form to reduce the visual details in order to increase the performance without compromising the quality [James 1990]. The 3D engine is usually a wrapper with a given 3D API such as OpenGL or Direct3D and abstracts the so-called rendering pipeline. The rendering pipeline itself is implemented on the 3D graphics accelerators. These are graphics cards, which are included in nearly every computer.

The rendering pipeline is filled with 3D data, which can be changed while running through the pipeline. The result of the pipeline is a complete picture on the screen. The major part of the work will be done on the hardware. In earlier days graphics

cards provided only rasterizing (drawing several pixels on the screen). The graphics card calculates the transformations and illuminations of 3D objects.In order to get rid of programming directly on the 3D hardware the 3D interfaces OpenGL and Direct3D are used which execute the desired operations on the hardware, although these operations can be emulated by software. The exact interface between the 3D API and the hardware is the 3D driver, which is generally provided and updated by graphic card developers.

2.3.5 Scene Graph

As aforementioned, the main task of a 3D engine is to maintain and display 3D objects. An easy but at the same time very poor solution is to put the objects in an interlinked list to display all of them one by one on the screen. Due to the high amount of 3D objects in the virtual environment it is not possible to get a quick screen layout of the whole scene –even objects which will not be seen would be processed. However, the graphics accelerator does not know which objects can be seen and which cannot. That is the reason why it is called low level [Cunningham 2001].

The virtual environment with all its 3D objects is called a scene and relating these objects to each other in a hierarchical order is called the scene graph. In virtual environments, this scene graph is equal to a tree with a root node. Every element can have multiple child elements but only one parent element.

2.3.6 Collision Detection

A fundamental problem in computer animation is collision detection. Interactions between moving objects are modelled by dynamic constraints and contact analysis. Some applications using collision detection are physically based modelling, geometric modelling and robotics. The principle is the same in all fields –especially computer games, which rely heavily on good collision detection. All areas are filled with virtual objects, so a character should stop when a door is locked and should not pass through other characters or objects [Lai 2009].

Collision detection, as used in the games sector, usually means intersection of any form. Intersection detection is the general problem. It means to find out if two

geometric objects intersect. A subtle point of collision detection is about the algorithms for finding collisions in time as much as space.

In computer games, the aims of collision detection are different. A good collision detection algorithm has to determine if the player or character has hit a wall or obstacle to stop them walking through it. It has to determine if a projectile has hit a target and detect points at which behaviour should change. For example, a car in the air has to return to the ground. Another task is to clean up animation. For example, to make sure a moving character's feet do not pass through the floor.

The geometry of the colliding objects is the primary factor in choosing a collision detection algorithm. An object could be a point or a line segment. An object could also be a specific shape such as a sphere, triangle or cube. These specific shapes can also be concave or convex, solid or hollow, deformable or rigid, manifold or non-manifold.

2.4 Rule-Based Systems

Rule Based System can be defined as any system that utilise production rules, in any structure, which can be applied to data to produce results; for example in simple systems that form validation and dynamic expression. Rule-Based Systems and dynamic scripting allows us to adapt code dynamically in mobile environments where users can create content and make changes to content in virtual environments. Simple rules allow average users to create natural language interfaces to interact in those environments. Different techniques have been developed, for example, to combine static and dynamic analysis to find out phrase structure and access semantic information. Combining rules systems with database systems to access information smoothly, reduces rule-processing costs and saves limited bandwidth. Semantic information is used to provide intelligent caching. This approach is much better than the traditional approach supported by trace-driven simulation results which successfully demonstrate its practicability and potential [Wu 2008].

Rule Engines use technologies like expert systems, decision tree, Genetic Algorithms, and neural networks to perform intelligent tasks. Rule based systems

implement knowledge and perform reasoning. Expert Systems use knowledge representation to arrange information in a knowledge base and knowledge base systems are known as applied artificial intelligence. Knowledge Engineering is a process of developing an expert system. The first developed expert system was called "Shells". The problem with early expert systems was hard coding logic but the "Shells" system was the first to separates logic from the system.

Drools [Drools 20011] can be best defined as a production system, which uses a Rule Based approached to implement an Expert System. The Rete algorithm [Forgy 1982] is used to evaluate rules in a working memory. The Expert System, which validates and evaluates expressions, is called a Production System as shown in Figure 2.2. A Production System focuses on knowledge representation using propositional and first order logic in a concise, none ambiguous and declarative manner. An Inference Engine is the computation part of the Production System that can scale to a large amount of rules and facts. The inference Engine matches facts and data against production rules and the process of matching the new or current fact against production rules is called Pattern Matching. In Production Systems the Inference Engine performs pattern matching using different algorithms such as linear, Rete, Treat and Leaps.

Figure 2.2 A Basic Production System [Proctor 2008]

Drool has a simple GUI based interface, which lets you write rules in natural language. The Rete Algorithm is used to evaluate rules [Bellifemine 2008]. The

implementation of the Rete algorithm in Drools is called ReteOO. The Rete Match technique was developed in 1974 to match patterns to a set of objects in order to evaluate all the possible outcomes [Forgy 1982]. The Rete Match Algorithm uses a tree structure-sorting network or index for the patterns matching to avoid iteration over the patterns. This makes it efficient if large volumes of patterns and objects are used. State information is used to avoid iteration over data and when it enters the data memory it starts computing partial and full matching for each object and keeps the object information until it has been removed from memory.

Drools uses an enhanced and optimized version of the Rete Algorithm for its object oriented system. The way Drools stores Rules in the Production Memory is illustrated in Figure 2.3. Facts are put in Working Memory and these are used by the Inference Engine. Facts are forced to stay in Working Memory where they can be changed or retracted. If a system has a large number of rules and facts, many rules could be true and conflict. The Agenda Manager executes the order of conflicting rules using a Conflict Resolution Strategy.

Figure 2.3 Forward Chaining [Proctor 2008]

Drools only supports Forward Chaining, which is "data-driven" reasoning. Facts are asserted into working memory which results in one or more rules being simultaneously true and scheduled for execution by the Agenda as shown in Figure 2.4. Forward Chaining starts from fact and propagates until the conclusion is reached.

Backward Chaining is "goal- driven" reasoning and starts from the conclusion and works backwards towards facts. If it cannot execute the conclusion, it will find a sub goal which will help satisfy some unknown part of the current goal. This process will continue until either it proves the initial conclusion or there are no more sub goals as shown in Figure 2.4.

Figure 2.4 Backward Chaining [Proctor 2008]

2.5 Dynamic Scripting

The word Script came from the performing arts where a human speaks a dialog to perform his/her arts. In computer science the term script means a piece of code that can be inserted or attached to an application at run time either to enhance the functionality of that application or plug in new components at run time [Loui 2008]. Scripts are directly interpreted from source code or byte code thus they are called an interpreted language. Initially Scripting languages were called batched languages or job control languages. The first script language "Shells" was developed in 1960 to enable a time-sharing system to avoid re-inputting a sequence of commands from computer terminal keyboards. The revolution of the WWW (World Wide Web) in the mid 90s has lead to more complex scripting languages with more powerful functionality. Languages like Perl often used to hold the web together. With the help of Netscape and Sun Micro Systems, new advanced scripting languages have been developed; JScript for Netscape and Microsoft VB Script (Visual Basic) for IE (Internet Explorer). In Web computing two types of scripting language are used: one that runs on the server and the other runs on the client side For example, on the client side Java Script could be used and on the server side scripting languages like PHP (Hypertext Preprocessor), ASP (Active Server Pages) and JSP (Java Server Pages) could be used.

The most important features of scripting languages are that they can enhance or create new functionality at run time. In virtual environments, it's important to extend functionality and allow users to contribute content creation and modify existing functionality or animations within the environment or a single object. There are numerous reasons for incorporating scripting languages [Ousterhout 1998] into virtual environments, the productivity benefits, the separation of data and logic, the scalability of the architecture, and the ease with which it can be achieved means that scripting languages have become part of a programmer's toolkit. Scripting is best utilized for the control of data and behaviour, and as can be seen by the numerous commercial titles that have chosen to script. There are many non-technical benefits over hard-coding, such as the flourishing modding communities surrounding games of varying genres.

There are many options for incorporating scripting into games and networked virtual environment applications, by utilizing an existing solution, or developing a custom bespoke tool. Bespoke tools can be very powerful, and they can provide the developer with the ultimate control over the solution, but involve an extremely large amount of development to create them from scratch. This puts this method in conflict with the productivity gains made from utilising scripting. When adopting a generic scripting language approach, all four of the languages mentioned provide an adequate range of language features, high-level bindings to other languages, built in support functionality, and portability. The choice will be application dependent e.g. using Lua if speed and memory footprints are crucial, Python if existing functionality must be utilized, or Ruby for maintaining an Object-Oriented approach.

JavaScript is a scripting language, which means it can be interpreted instead of compiled, and executed at run time. JavaScript was developed by Netscape for the Netscape browser to compete with the Microsoft scripting language VB and to include in the browser rich website interfaces [Kirda 2009]. One advantage in our work is the use of JavaScript in the virtual environments to dynamically generate behaviours. JavaScript is one of the most popular scripting languages for embedding source code into an object without the need for compilation, making it easier for users to write their own source code or modify existing code.

2.6 Peer to Peer Networking

P2P refers to a network that enables two or more peers to share or exchange information without involvement of any central infrastructure. File-sharing systems such as Kazaa [S.Networks 2008], Napster [DeVoss 2006], and Gnutella [Gnutella 2008] are considered some of the best examples of P2P networks. IM (Instant Messaging) is another example of a P2P. A P2P application offers advantages in contrast to other architectures such as client/server, such as performance, persistence, and cost. P2P operates independently of any central coordination, which enables users to share or exchange information without worrying about the location of the peer. For example, in the case of Kazaa, which is mainly used for music file sharing, users search for a particular song and download it without having to be aware of the location of the host peer.

P2P is classified based on the connection and routing methodologies they use. The most common models are; the Centralised P2P model, Pure P2P model and Hybrid P2P model. In the Centralized P2P, model peers connect to one or more servers to locate other peers. Once peers have been discovered they communicate between each other without the use of a central server. IM is an example of such a model. The pure P2P model does not use any centralized server for peer discovery. It relies on the cooperation between peers by exchanging location information between them. A peer would usually use the location information gathered from previous connections to connect to the network and to inform the rest of the network about its existence. Devices look for bootstrap servers when they connect to the network. Gnutella is a good example of such a model. The structured P2P model gains location information by cooperation with peers and the use of previous knowledge of the resource location. They do not rely on central indexing servers but on the knowledge gained from previous participation in the network, System like PAST [Druschel 2001], and Scribe [Castro 2002] use the hybrid P2P model.

Underpinning any solution will require a more flexible communication infrastructure. Perhaps with the advent of IPv6 we will see a return of a true peer-to-peer Internet because a large address space will remove need of NAT, much as it was during its early infancy. This is an interesting way forward where the obvious benefits P2P applications afford have become widely evident over the last five years. This is especially true in the multimedia and entertainment environments where television corporations now utilise P2P as an effective distribution medium and where massive multi-player online games can only truly realise their full potential through P2P adoption. Here we discuss a number of important research areas we hope to build on.

Nitin Desai *et al.* [Nitin Desai 2003] introduces Konark, a service discovery and delivery protocol for ad hoc P2P networks in which the authors provide an infrastructure to set up generic peer-to-peer systems. It acquires advantages of basic networks for peer naming and message routing. It uses entirely distributed, peer-to-peer techniques for resource discovery which provides every peer with the ability to publicise and discover the resources in ad hoc networks [Nitin Desai 2003].

We also see considerable efforts within traditional networks where multimedia solutions on the Internet have seen widespread adoption, which include voice over IP [Vilei 2006; McLaughlin 2007], IP TV [Luo 2007], and P2P content distribution models from organisations such as the BBC with their iPlayer (previously iMP) implementation. Research initiatives are also well underway to make television more life-like. Fehn *et al.* [*Fehn* 2006] provide an interesting account of 3-DTV where 3D representation formats, auto stereoscopic 3D displays, multimodal interfaces, personal 3-DTV, and group viewing are discussed. Different manufactures such as Panasonic, Sony and others are offering 3D TV but one difficultly with the majority of these devices is that they require viewers to wear glasses to watch TV. Our framework also provides mechanism for streaming 3D content for use with interactive TV.

In Massively multiplayer online game (MMOGs) distributed P2P mode uses "Coordinator" nodes for managing lesser groups of players, this has been particularly effective for supporting MMOGs [Knutsson 2004]. The ability to self-organizing of P2P networks allows them to dynamically balance the increase and decrease of players. The game shares CPU cycles and memory to handle game state but this is limited to the duration of game play.

2.6.1 P2P Models

In this section we discuss P2P Models in detail which can be classified into three types on the basis of the connection and routing methodologies they use; the Centralised P2P model, Pure P2P model and Hybrid P2P model.

In the Centralised P2P model illustrated in Figure 2.5 peers, connect to one or more servers, in order to locate other peers. Once peers have been discovered, the communication between peers is carried out without use of the central server. In Centralised P2P networks the resources are indexed on a central service and peers query this server to locate peers with the desired resource. The Napster [Nagaraja 2006] and IM applications are examples of such a model, where connection information is retrieved using a central server but connections are made directly with peers. One of the advantages of such a model is that resources can be located quickly

and efficiently; which is beneficial to monitor users, as they would be registered in the system. On the other hand, this system is prone to failure mainly 'single point' of failure. In addition, the system can be susceptible to 'denial of service' attacks as servers have to cater for all the requests made, which affects the rest of the network.

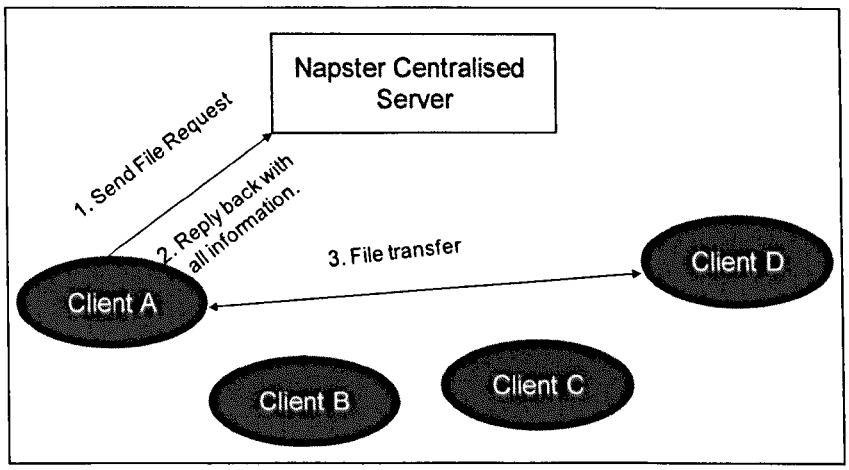


Figure 2.5 Napster's centralised P2P model

The Pure P2P model showed in Figure 2.6 does not use any centralised server to assist in peer discovery. Instead, it relies on the cooperation between peers by exchanging location information between them. Peers connect to the network and use location information gathered from previous connections for the discovery of other peers and to inform the rest of the network about its existence. Applications using this model initially look for a bootstrap server when they join the P2P network for the first time [Gauthier 2008]. Gnutella [Gnutella 2008] and its successors are classified as pure P2P networks. One of major advantage of being completely decentralised also is to avoid a single point of access and failure, and thus is fault-tolerant. However, it can be slow and traffic intensive, as they tend to be insufficient in the lookup process. There is also an issue with the lookup horizon in unstructured P2P networks, where the resource may be available on the network but the lookup process cannot locate it. Scalability is not an issue in the pure P2P model but the lack of guarantee of finding resources can be a shortcoming of these systems. This model works best for systems where resource location is not of highest importance such as IM. This model has a low cost of entry and maintenance.

Figure 2.6 : Gnutella pure P2P model [Science 2006]

The structured P2P model gains location information by cooperation between peers as well as previous knowledge of the resource location. They do not rely on central indexing servers but on the knowledge gained from previous participation in the network, e.g. systems like PAST [Druschel 2001] and Scribe [Castro 2002]. The location of the resource is related to the resource name by located by regularly querying the network and updating routing tables. These systems overcome the issue of limited lookup because of the nature of the underlying routing protocols, which guarantee the discovery of resources. These routing protocols include Content Addressable Network (CAN) [Ratnasamy 2001], Chord [Balakrishnan 2003], Kademlia [Maymounkov 2002] and Viceroy [Malkhi 2002], which rely on the use of the a Distributed Hash Table [Takeda 2008] abstraction as a method for lookup and data location. This model works best for system where resource location is of highest importance.

2.6.2 Name Based Classification

A P2P network [O'Mahony 2003] consists of all peers as network nodes; there are links between any two nodes in the network i.e. a participating peer knows the location of another peer in the P2P network. This leads to two types of network: Structured P2P networks [Hsiao 2003; Lua 2005] and Unstructured P2P networks [Lua 2005].

Unstructured P2P networks are formed when the overlay links are established randomly. Such networks can be established easily when a new peer joins the network, copies links of another node and then forms its own links over time. In unstructured P2P networks if a peer wants to find data over the network, the query is flooded through the network – mainly used by file sharing systems where search might take a long time to respond. Flooding causes high traffic in the network results and poor search efficiency. It is difficult to find specific files if the content is low in popularity i.e. as shared by few peers then there is more chance that the search will be unsuccessful. Popular P2P networks such as Gnutella [Chawathe 2003] and KaZaa [KaZaA 2008] are good examples of unstructured P2P networks.

Structured P2P networks overcome the problems of inefficient routing and the inability to locate rare objects in unstructured networks by using the Distributed Hash Table [Balakrishnan 2003; Takeda 2008] technique and allowing each peer in the network to be responsible for a specific part of the content. These networks use a hash function to assign values to every piece of content and then follow a global protocol identifying which peer is responsible for which content. When a peer wants to search for specific content it first determines, using a global protocol, which peers are responsible for that data and then directs the search towards these peers. These network guarantee to locate objects [Balakrishnan 2003], CAN [Chen] Pastry [Druschel 2001] are well known examples of structured P2P networks.

2.6.3 P2P Applications

In this section, we discuss some application of P2P networks.

1. Academic search engine, the Sciencenet [Liebel-Lab 2008] provides a free search engine for scientific knowledge. It is based on YaCy technology [YaCy Distributed Web Search 2011], based on P2P principles, written in Java. Using Sciencenet educational institutes can contribute their own peers to encourage them to contribute to the scientific network.

2. Many organisations are trying to apply P2P networks for educational purposes. LionShare enables academic users to search and retrieve contents from other LionShare users. LionShare only allows login with university access accounts and does not allow any anonymous sharing of files.
3. US military Department of Defence has also started research projects based on P2P networking technology for modern warfare strategy [Walker 2001]. However, details regarding these systems are kept classified due to security reasons.
4. A number of P2P applications are used to deliver TV content over the internet such as P2PTV.
5. P2P networks are widely used in business not only in file sharing but also in distributed computing, eMarketPlace [D'Aubeterre 2009] and office automation.
6. Due to an increase in demand for voice and video conferencing in real-time, telecommunication is now also using P2P networks [Jennings 2006]. Skype is one of the well-known used Internet application for voice communication also based on P2P technology.

2.7 Summary

This chapter discussed the history of virtual environments and various games and social networking technologies such as Second Life. We considered how the virtual environment started in the form of text based games in the 60s but later on moved on to using more advanced graphical formats due to advances in computer technologies. The chapter also discussed Rule-Based systems and their role in mobile environments where logic is separated from application. The chapter has also highlighted how scripting provides a better way of writing plug-ins for an application at runtime.

In the final section of this chapter P2P networking has been discussed in detail and we highlighted the advantages of P2P over server-client architectures. Different P2P models have been discussed in this chapter such as the Centralised P2P model, Pure P2P model and Hybrid P2P model. Various types of P2P application were also discussed.

Chapter 3

3 Ubiquitous Computing and Networked Appliances

3.1 Introduction

This Chapter provides an overview of the work carried out in the main research areas relevant to this thesis, which includes Ubiquitous Computing and Networked Appliances and Home Networking,Cutting edge research initiatives are highlighted including their associated limitations, which are addressed within this thesis.

3.2 Ubiquitous Computing

Ubiquitous Computing was first defined by Mark Weiser [Weiser 1991]. A great deal of research is now undertaken to investigate the use of small devices embedded within the environment that are capable of transferring and receiving information [Tsung Teng 2008]. Ubiquitous Computing offers technology anytime, anywhere and to anyone [Hong 2009]. Context Awareness is important because it provides information about the status of people, places, things and devices in the environment. Context can be derived from information that can be used to distinguish the situation an entity, such as person, place or an object that is considered relevant to communications between an application and user, which includes location, time, activities and the preferences of each entity. A context-aware system is capable of extracting or interpreting and using context information to adapt its functionality to the current situation use. The purpose of context awareness [Bricon-Souf 2007] is to provide services to particular places and people.

In the consumer electronics market users are familiar with and have become increasingly more dependent on wireless communications. However, it is only now that we begin to see solutions beyond simply Internet connection sharing or ubiquitous digital content accessibility. Sensor networking, networked appliances, and

home networking have now matured to support real-world artefacts [Merabti 2008a] further moving us towards Weiser's vision [Weiser 1991]. For example, rooms that describe themselves and the artefacts they contain, to visually impaired people when they enter the room in the virtual world feels like a real world room.

Whilst this is very beneficial, several difficult challenges remain, e.g heterogeneity and interoperability. One technique used to help elevate these challenges is to use a virtual container to simulate ubiquitous computing environments [Jinseok 2005]. Test beds like this have been used to validate the degree of interaction, usability and simplicity in software testing platforms and above all to determine a means of mapping or creating a bridge between the virtual container and physical devices.

Using a more proprietary and focused solution learning environments have also allowed the users to enjoy the benefits of ubiquitous computing to enhance better learning experiences. For example, the SULOMS (Semantic-Oriented Ubiquitous Learning Object Model) [Lili 2008] is used to meet the needs of teachers and students using mobile devices, irrespective of location, to accomplish the basis for multi-mode courseware and its semi-automatic generation. The general idea being that the delivery of teaching material and its consumption can be performed in a more mobile and ubiquitous way without having to bind users to physical locations.

Perhaps the most compelling area where the physical meets the virtual is within the field of bio-mechanics and motion capture [Tong 2004] Many applications have been proposed, from entertainment (special effects) to health (rehabilitation) [Zhang 2008]. Whilst traditionally such systems have been expensive and are typically confined to dedicated facilities, relatively cheap wireless sensors have now made it possible to embed such technology within the home [Zhou 2008]. This has been particularly important in the area of rehabilitation allowing information about gait change, joint angles and movement to be collected for analysis [Fergus 2009]. Sensors attached to the body are mapped to virtual environment avatars where physical movement is used to control motion. Xbox Kinect [Xbox Kinect 2011] is developed by Microsoft which allows user to play games without physical controller by using motion sensor that

track entire body of the person. Other console such as Wii [Nintendo 2011] and Play station [Play station 3 2011] uses physical controller to play games.

In the following subsection, we discuss some of the existing research projects in ubiquitous computing and their limitations.

3.2.1 Aware House project

With greater flexibility, engines now provide advanced tools to allow a bridge between real and virtual environments. For example, the Aware House project [Cory 1999] has developed a completely augmented home in which ubiquitous devices embedded within the environment can be controlled via a graphical virtual world - switching a light on in the virtual world results in the light being switched on in the real world [Brian 2009].

A limitation with the Aware House project is that it needs to be maintained and developed over time and requires human effort to add or remove devices. It does not allow objects to be shared across different virtual environments or for behaviors to be generated dynamically.Users are increasingly becoming empowered to influence digital entertainment, providing a platform for interactive multimedia content sharing across different virtual environment is not possible in Aware House Project.

3.2.2 Sixth Sense

Sixth Sense is an MIT research project to augment the physical world with digital information using wearable gesture interfaces [Pranav 2009]. Sixth Sense research focuses on bridging the gap between the physical world and digital information to help support decision making in our day-to-day living, which is not always possible using our five natural senses. Advances in technologies have shrunk the size of computers to handled devices that can be carried in our pocket such as mobile phone which have processing capabilities equal to desktop computer five or ten years ago. This has made it easy to connect people anytime and anywhere, however there is no link that exists between the physical world and digital environments. Sixth Sense uses natural hand gestures to interact with virtual worlds and frees information from its

limitations by seamlessly integrating it with reality, and therefore making the physical world a digital device.

The Sixth Sense prototype consists of a cameraand pocket projector connected to a mobile device. The camera used in the prototype is used to identify and track the user's hand gestures and physical objects using computer vision based techniques. The projector projects visual information to enable surfaces, walls and physical objects around us to be used as interfaces. The video stream data captured by the camera is processed by software programs to track the location of the coloured markers at the tip of the user fingers using computer vision techniques.

The Sixth Sense has many advantages and is used in a variety of applications such as map applications that use nearby surfaces and hand gestures, similar to gestures supported by Multi-Touch based systems, letting the user zoom in, zoom out or pan using intuitive hand movements. The drawing application allows users to draw on any surface by tracking the fingertip movements of the user's index finger. However, Sixth Sense has several limitations, - devices have to be explicitly introduced and programmed instead of being dynamically incorporated on the fly. Digital information with other virtual worlds is not possible in Sixth Sense.

3.3 Networked Appliances

In the following sub-sections, we discuss some of the more common standards used within industry and academia alike to interconnect networked appliances within the home.

3.3.1 Digital Living Network Alliance project (DLNA)

The establishment of DNLA in 2003 was to enable cross-Industry convergence of multimedia content in home networks [DLNA 2004a]. The main goal was to enable a wired and wireless interoperable home network where digital content in the form of video, music and images can be seamlessly shared across personal computers, consumer electronics and mobile devices. DLNA allows seamless and effortless sharing of multimedia content across different hardware platforms, for example, streaming a video clip wirelessly from the phone to the TV. DNLA simplified file sharing so consumers do not have to concern themselves with how content is stored.

Although DNLA allows content to be shared this is restricted to the passive sharing of content like video files, images or audio - it does not allow content to be actively shared, such as game objects that require their behavours to be dynamically created as posited in this thesis.

3.3.2 Open Services Gateway Initiative (OSGi)

OSGi was founded in March 1999, to develop an open specification for the delivery of managed services in LANs. Service providers, developers, and gateway operators use OSGI to develop, deploy and manage services in WANs and LANs [OSGi 2009]. OSGI enables service providers such as cable operators to deliver services over the network to devices that implement the OSGi specification. For example, mobile phones, PCs (Personal Computer), and consumer appliances.

OSGi provides a general purpose Java framework that supports their deployment of downloadable and extensible applications called bundles. Devices that support OSGI specification can download and install these bundles and remove them when they are no longer required. Bundles can be dynamically installed and updated on any platform with varying capabilities. It is capable of hosting multiple applications from different service providers or single services platform. It also allows service deployments that provide functionality to other services. The OSGi architecture comprises of number of entities, a services platform, and an application server connected to both WAN and LAN. User service providers that deliver applications to gateway operators verify users prior to delivering services. The Gateway operator is responsible for the operation of the gateway. It offers service providers a secure environment to execute services. The operator is responsible for installing and removing applications in order to ensure that sufficient resources are available. It also monitors the gateway to detect any security attacks.

In most cases OSGi is managed and controlled by a centralised controller owned by service providers, howeverconfiguration by users requires some technical knowledge. OSGi does not provide a way to dynamically generate behaviour for a

device started or connected to the network. OSGi is a low level protocol thus does not provide the functionality to visualise devices in virtual worlds or to control functionality.

3.3.3 Reconfigurable, Ubiquitous, Networked Embedded Systems (RUNES)

The RUNES [Koumpis 2005] project has provided a framework that allows any device to form part of the Internet irrespective of its capabilities. The RUNES approach to development is to provide a standardised software framework that employs a uniform "Software Component" abstraction which can be implementable on various types of devices [Costa 2005]. This is seen as an important step where advances in sensor networking technologies have emerged to be new sources for multimedia data.

RUNES claims to provide a mechanism in which it would be easy to configure, deploy and dynamically reconfigure network-embedded software. However, it does not provide a way to control and visualise devices in virtual worlds.

3.3.4 UPnP

With the introduction of sophisticated devices and high speed Internet connections consumers can do much more with their network, such as store and stream data, and use mobile phones to access or transfer media files. However, due to cost and complexity, installing a network to provide such activities makes this idea impossible or beyond the reach of non-technical users. Such users are looking for an affordable and easy to install solution. UPnP [UPnP 2006] has emerged as an industry consortium to create a "plug and play" solution. The UPnP forum is comprised of more than 810 member companies [UPnP 2006] across many industries. CE (Consumer Electronics) manufacturers within this forum develop products that incorporate the UPnP technology [Microsoft 2004]. UPnP is a set of protocols that allows devices to advertise their services; these services can be discovered by other devices in the network. The UPnP Implementers Committee (UIP) was formed in 2001, to promote the adaption of UPnP technology, which also administrates the UPnP device certification. UPnP certified devices range from printers to mobile control devices. UPnP technology can run on any medium that supports IP networking

including phone lines, and power lines. Product vendors can use any programming language and operating system to build UPnP products. UPnP technology is built on IP, TCP, UDP, HTTP and XML.

UPnP supports zero-configuration i.e. any vendor device can dynamically join the network, obtain an IP address and advertise its services without any restriction. Devices can also leave the network at any time. However, UPnP cannot access services outside the local network; all communication is performed over IP, and devices must obtain an IP address before joining the UPnP network. By using an IP address with service discovery performed using IP broadcast, a control point can only contact other devices within the same subnet [UPnP 2006]. Messages within UPnP are sent using SOAP [Louridas 2006]. Each device must have a DHCP (Dynamic Host Configuration Protocol) [Wu 2007] client and search for a DHCP server when it first connects to the network.

3.4 Summary

This chapter discussed different home networked appliances and ubiquitous computer to allow devices to be interconnected and visualise them in virtual worl. Solution such as Aware House or sixth sense allows physical devices to be visualised in virtual world in greater flexibly to bridge between physical and virtual worlds.

Different networked appliances solutions are discussed in this chapter such as OSGi and UPnP in attempt to integrate devices, but they are managed and controlled through centralised providers. However solution introduced are low level protocols which does not provide a way of visualizing devices and controlling them from virtual environment where they can be discovered and generate their functionality on the fly.

3.4.1 Challenges

This Chapter has described the key research within the areas of Networked Appliances, and Home Networking. We have identified several key challenges pertinent to this thesis that have not been addressed in the above-mentioned approaches. Each of these challenges are listed below and addressed throughout the remainder of this thesis.

- The first Challenge is to develop a system to allow us to bridge the gap between physical and virtual worlds. This will require new algorithms that link devices and virtual artefacts as well as moving them between heterogeneous environments. There is a need to create artefacts that are modular and descriptive in terms of their appearance and their behaviours.
- The second challenge is to develop services that need to transcode 3D model data and dynamically support behaviour development using a combination of on-the-fly scripting and semantic reasoning using a logic system such as a rule engine [Maciol 2008]. This will improve interactivity and the way the platform allows manipulation and control of content, to the extent that new user-generated multimedia can be constructed purely through the combining and manipulation of 3D multimedia streams using networked appliances.
- The third challenge is to allow the platform to effectively generate functionality dynamically and allow us to map different object behaviours using ontologies.
- The fourth challenge is to allow the user to seamlessly move digital content between heterogeneous environments, where content from such environments can be shared. Typically, this is not the case where proprietary platforms host virtual environments with limited or no openness to include others. Given these limitations, we aim to provide an open platform on which any virtual environment can interoperate with each another and share, use, and create content in any of them – for example, content owned in Second Life, can be moved and used by users in The World of Warcraft.
- A final challenge is to develop protocols that bridge the gap between the two (virtual and real world). Building virtual worlds on top of a networked appliances platform that forms part of the Internet, is unique both in terms of its design, and the level of user control that the resulting platform will provide.

Chapter 4

4 A Framework for the Visualisation and Control of Ubiquitous Devices, Services and Digital Contents

4.1 Introduction

In Chapter 3 we presented a discussion on current Ubiquitous Computing platforms and highlighted their limitations, i.e. physical access to hardware, and invisible services in home networks. These approaches are not flexible and they do not provide any way of generating behaviours on the fly. They provide limited control over devices within the network and the services they provide. They do not provide any mechanisms to effectively distribute services across the network or discover those services using high-level semantics.

In addressing these limitations this chapter presents the Visualisation and Control of Ubiquitous Devices, Services and Digital Contents (VCUDSDC) Framework. More specifically, the framework addresses the challenges discussed in Chapter 1, which include behaviour discovery; behaviour matching; dynamic behaviour generation, and ubiquitous computing. The framework allows devices to be visualised and their behaviours to be projected into a virtual environment. Functions provided by devices can then be controlled and visualised in virtual environments – this provides more control for devices and hidden services that may not be possible physically. The framework also allows objects from different virtual and physical environments to be shared and visualised and to discover its behaviour using high-level semantic behaviour. The following section provides an overview of the proposed framework.

4.2 Framework Overview

The Visualisation and Control of Ubiquitous Devices, Services and Digital Contents framework is based upon the service integration framework developed within the Networked Appliances Laboratory at Liverpool John Moores University [Merabti 2008a]. VCUDSDC extends this framework to support additional framework services that address the requirements defined in Chapter 1.

Using the principles of service-oriented computing [Bichier 2006], components, such as game consoles and mobile phones, implement a small footprint of code allowing functions, such as audio, video, and gaming components to be disseminated within the network. Using the Service Integration framework [Merabti 2008a] services and components are linked to the network using any communication protocol. This allows them to discover and/or publish and use framework and application services locally (provided by the component itself) or remotely (provided by other components). Furthermore, it allows devices to perform semantic interoperability between different vocabularies used by component manufacturers; automatically form communication links with other components in the network; self-manage links with other components in the network; and self-manage their operations based on composite and environmental changes. Application specific services, on the other hand offer a means of dispersing and utilising component functionality, such as audio and video, gaming engines, player (AI behaviours) and game objects (tree, car or avatar).

This is achieved using the service integration framework [Merabti 2008a], implemented on every component – be it a networked appliance or a software module from the virtual world. This peer-to-peer interface can be mapped onto any middleware model. Devices connect to the network as either specialised components or simple components. A specialised component has the ability to provide services as well as to propagate service requests within the network. A simple component by comparison has more restricted abilities: it joins the network, propagates queries and invokes discovered services, for example, sensors in a network could be used to provide multimedia data for crowds or flocking. This enables any component irrespective of its capabilities to choose how it will interact within the network.

Using the basis of this architecture, we have designed a distributed service-oriented platform for use with virtual environments and physical devices. The architecture allows artefacts (real-device functionality such as audio, video and data from sensor networks; AI behaviours, and other non-player objects such as trees, rocks, and clouds) to be shared within and across different environments (both real and virtual).

It has been important to consider the overall structure that a virtual environment might take. The goal is to deconstruct as far as possible the holistic notion of an environment whether it is virtual or real, to a set of autonomous, generalised and reusable components.. The result however is considered from the opposite perspective. Ultimately, the aim is to allow artefacts to exist as an ad hoc interaction between various networked components, the entirety of which forms the virtual and real environment. None of these components in isolation can be considered one single environment itself. Perhaps the closest to what might be considered the heart of the environment might be the rendering or physics engines and behaviour enactment engines. However, these will only provide one of any number of interpretations of the interactions that occur between components. The overall architecture is illustrated in Figure 4.1 and a more detail discussion follows.

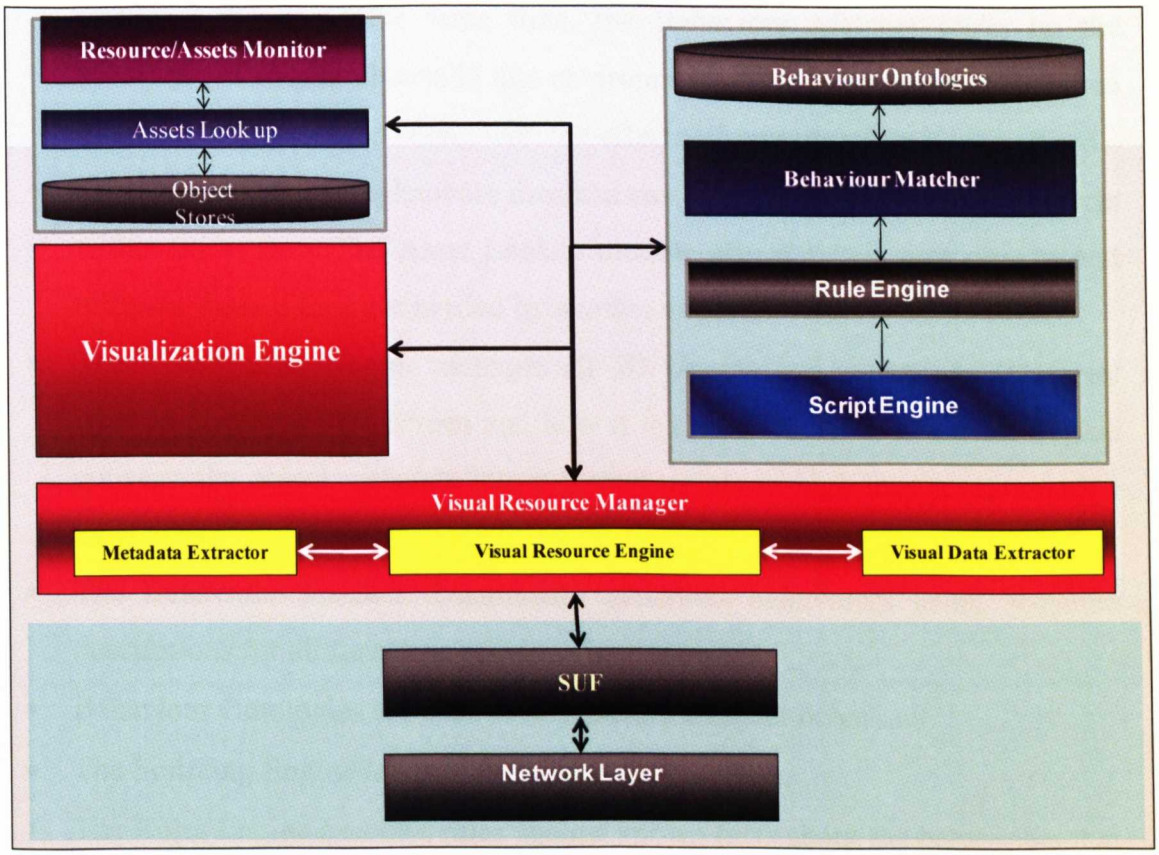


Figure 4.1 Visualisation and Control of Ubiquitous Devices, Services and Digital Contents Framework

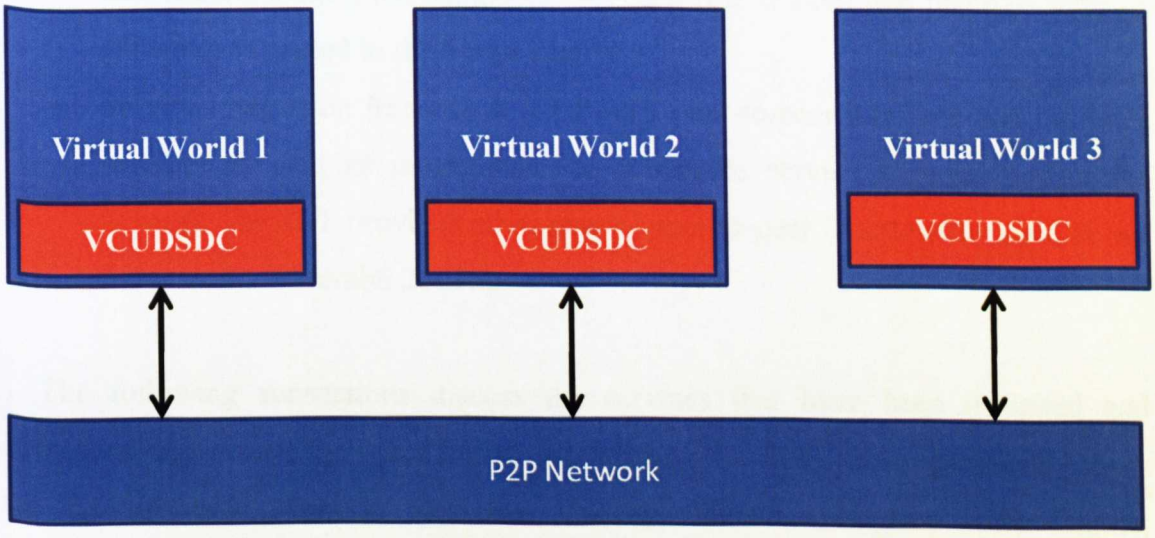


Figure 4.2 VCUDSDC Framework in Network

The architecture consists of the following

- The Visual Resource Manager registers objects with the Assets/Lookup. Following this, it extracts the meta-data associated with objects and passes it to the Visualization Engine, which in turn renders the 3D object into a

graphical shape. At the same time, the Behaviour Matcher looks up the behaviour of similar objects in that environment. If it finds the behaviour then it assigns it to the object along with the effects it supports when it is executed.

- The Asset Lookup implements mechanisms to register and discover objects within the system. The Asset Lookup module also registers new objects and retrieves them if they are needed by another system.
- The Visualisation Engine manages all 3D objects and scenes and interprets what to display on the screen and how it is drawn. The Visualisation Engine converts the object metadata into a format usable in high level programming languages.
- The Behaviour matcher dynamically generates behaviours using semantic descriptions for all functions provided by the object.
- Behaviour Ontologies are used to help describe object behaviour.
- The Scripting Engine loads script language compilers.
- The Rules Engine executes rules against known facts about the behaviours that objects support and dependent on matches, relevant rules are fired – these are referred to as Production Rules. Within the actions of fired rules script functionality is constructed that is known by the target environment and which best accommodates the behaviour. As each rule is fired, the piece of script it generates is passed to the Script Engine.
- Service integration framework (SUF) is a peer-to-peer interface that provides services as well as mechanisms to propagate service requests within the network The SUF provides all necessary peer-to-peer functionality used in our framework. [Merabti 2008a].

The following subsections discuss the services that have been designed and developed that extend the functionality provided by the SUF, more specifically these are the Resource Monitor, Resource Lookup, Metadata Engine, Rule Engine, Scripting Engine, Behaviour Matcher, and the Visualization Engine. The overall framework is depicted in Figure 4.1.

The component sequence diagram illustrated in Figure 4.3 illustrates how components interact with each other and pass information between the different layers

to generate and find known behaviours. This provides a snapshot over time that describes the relationship between objects and the information flows.

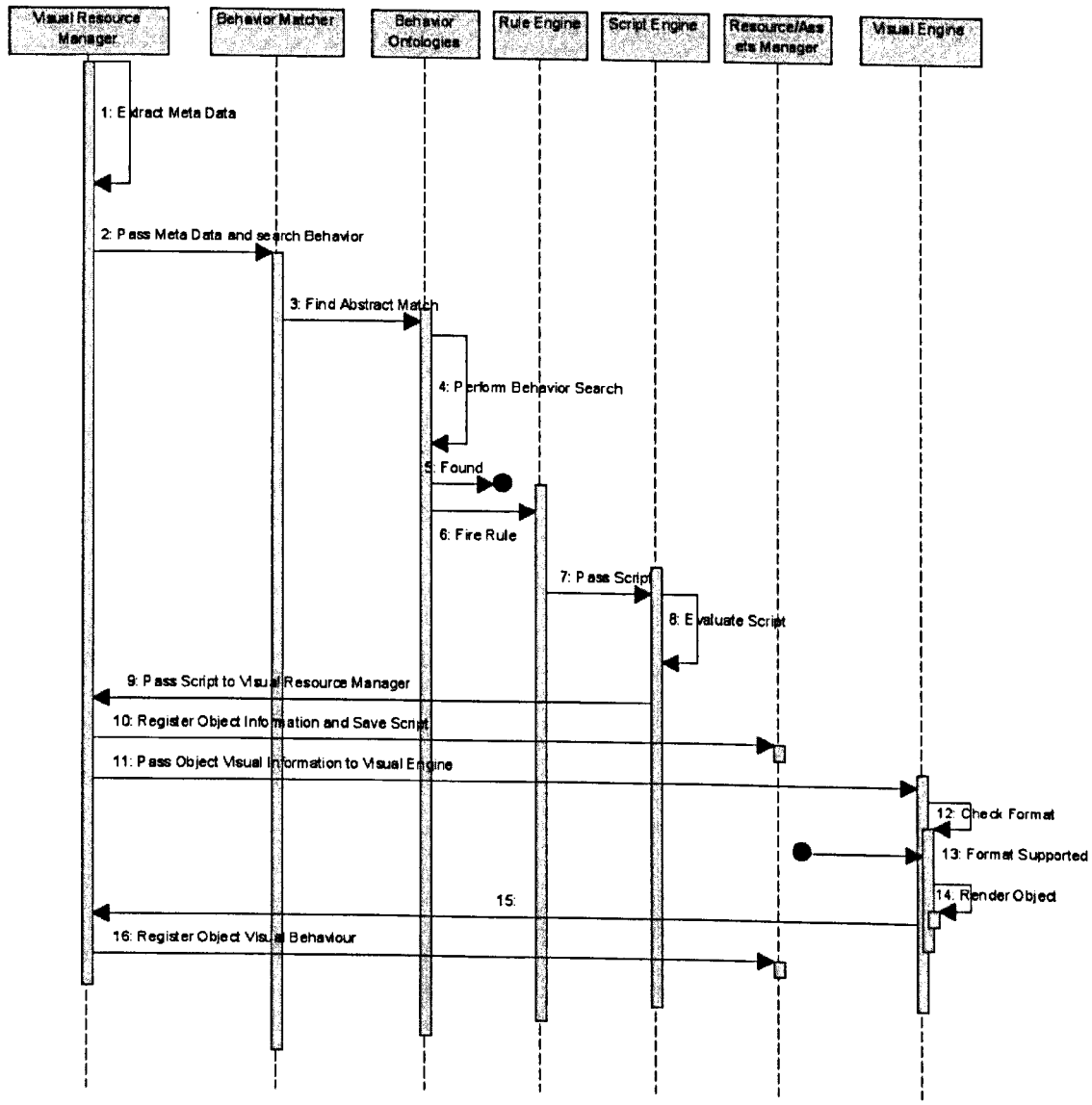


Figure 4.3 Component Sequence Diagram

In the first step the system interact with the object and extract meta data and using Ontologies in step 2 the system pass object description to behaviour matcher which peform abstract match using SUF. This allows us to start finding behaviour using Ontologies to find the behaviour. In step 5 when behaviour found the system fire rules to generate that behaviour and pass it to the script engine to validate the script in step 8, if the script does not have any syntax error the system attach it to object in step 9. In step 10 the system save the script to cached script. after script has been generated the next step is to render the object where system load the 3D model and pass that 3D model to Visual Engine in step 11 where it check the format of the 3D model in step 12 if format is supported then the system render the object in virtual world in step 14.

In last step the system registered the object detail after successfully generated the script and rendered the object.

In the following sections, we discuss the framework in more detail.

4.3 Visual Resource Manager Protocol

In this section, we will describe the Visual Resource Manager (VRM) protocol, which implements mechanisms to share and distribute and semantically discover objects and generate behaviours dynamically within heterogeneous environments (both virtual and physical). The VRM protocol contributes additional knowledge to this area by enabling environments to discover objects dynamically based on semantic matches between object requests and object behaviour descriptions as illustrated in Figure 4.4.

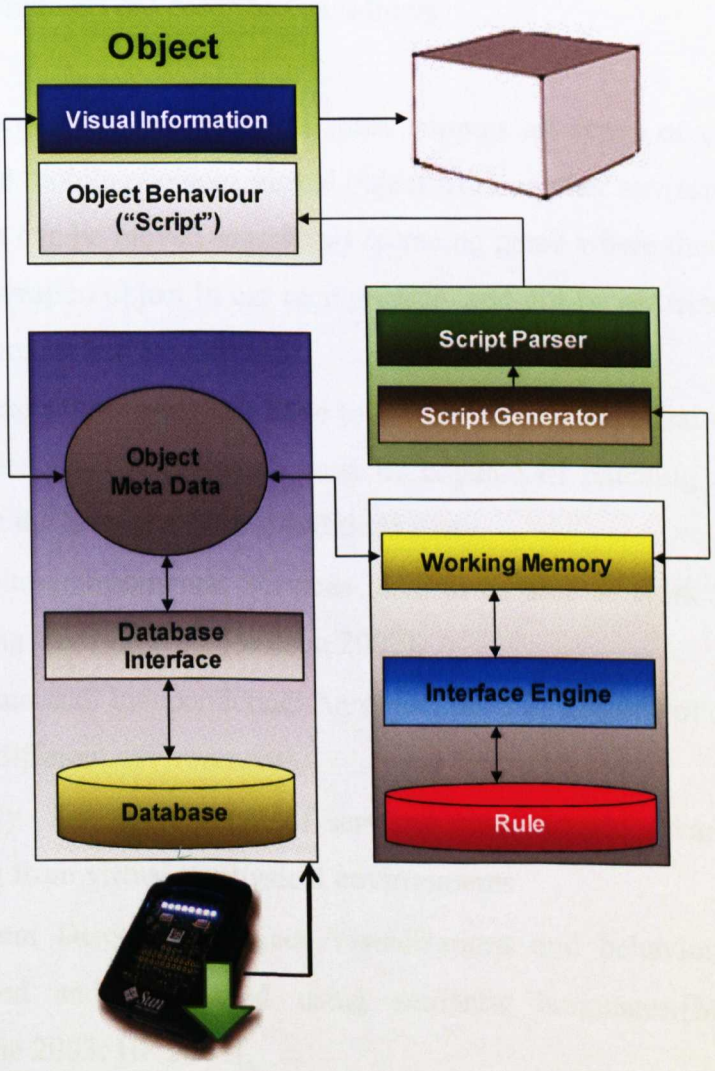


Figure 4.4 Dynamically generating object behaviours

After discovering new devices in the network, the virtual environment starts processing the meta data description and begins generating the behaviours using simple rules, which are then passed to the scripting engine to parse that script.

The following subsections describe the design requirements for the VRM protocol.

4.3.1 Protocol Requirements

Before objects can be effectively distributed and discovered, it is vital that the protocol addresses a number of key requirements. Implementing services within heterogeneous environments (both real and virtual) means that we have to make it generic, therefore the compatibility of objects with destination environments cannot be guaranteed. The challenge is to enable objects to exist in diverse environments and allow objects within these environments to effectively expose and discover these objects. The requirements are described as follows:

- Interoperability: The protocol must support all types of objects such as physical mobile phone or virtual object from another environment such as a weapon can be moved seamlessly to racing game where their is no concept of the weapon object in car racing game, and not be restricted to a specific environment like Second Life.
- Decentralisation: Services have to be completely decentralised; every peer that joins the P2P network must be capable of reaching any other peer without the help of a centralised third party.
- Dynamic environments: Services have to be able to work in dynamically changing environments [Wilson 2002].
- Environmental Independence: An object must be capable of communicating across different environments.
- Ubiquity: The deployment of services must include a variety of objects ranging from virtual to physical environments.
- Intelligent Discovery: Object Visualisations and behaviours need to be described and discovered using semantic languages.[McIlraith 2001; Maedche 2003; HP 2004].

- On-demand Objects: Objects will not run continuously and will be discovered and invoked as and when they are required [(TME) 2003].

4.3.2 Protocol Overview

The requirements described in Section 4.3.1 are addressed within the VRM protocol using four core components, which are described below and illustrated in Figure 4.1. The VRM protocol implements:

- A Visualization Engine to render objects.
- A Resource management component to track the use of resources and their usage (i.e. sharable or not).
- A Knowledge Base and Inference Engine to describe and process service descriptions and upper ontologies for semantic interoperability.
- A Behaviour container capable of dynamically generating and translating object behaviours

Functions offered by objects are exposed using services, which are described using behaviour ontologies that represent the capabilities supported by target environments. Invocation of behaviours is achieved using the behaviour interface, which describes the signatures it supports.

The knowledge base resides in each target environment and contains the behaviour ontologies and the ontology structures that allow environments to resolve terminology differences between different vocabularies. The behaviour ontologies and the semantic interoperability ontology perform different functions and do not represent a single ontology. Behaviour ontologies only describe the behaviours hosted by target environments. The semantic interoperability ontology is a small ontology that continually evolves over time based on peer collaborations, which ensures that the knowledge within the peer network is decentralised [Merabti 2008a].

A detailed design on these aspects of the framework is presented in the following sub section.

4.3.3 Protocol Design

The Activity Diagrams presented in this section describe the design of the VRM protocol. These diagrams illustrate how virtual environments perform the start-up

process; how behaviours are created; and how peer advertisements are created, published and discovered within the network.

When a virtual environment is initially switched on, it executes a start-up procedure to connect it to the network. The start-up process is shown in Figure 4.5. At start up it loads all stored objects from the data-source one by one and performs the object creation process using different attributes stored.

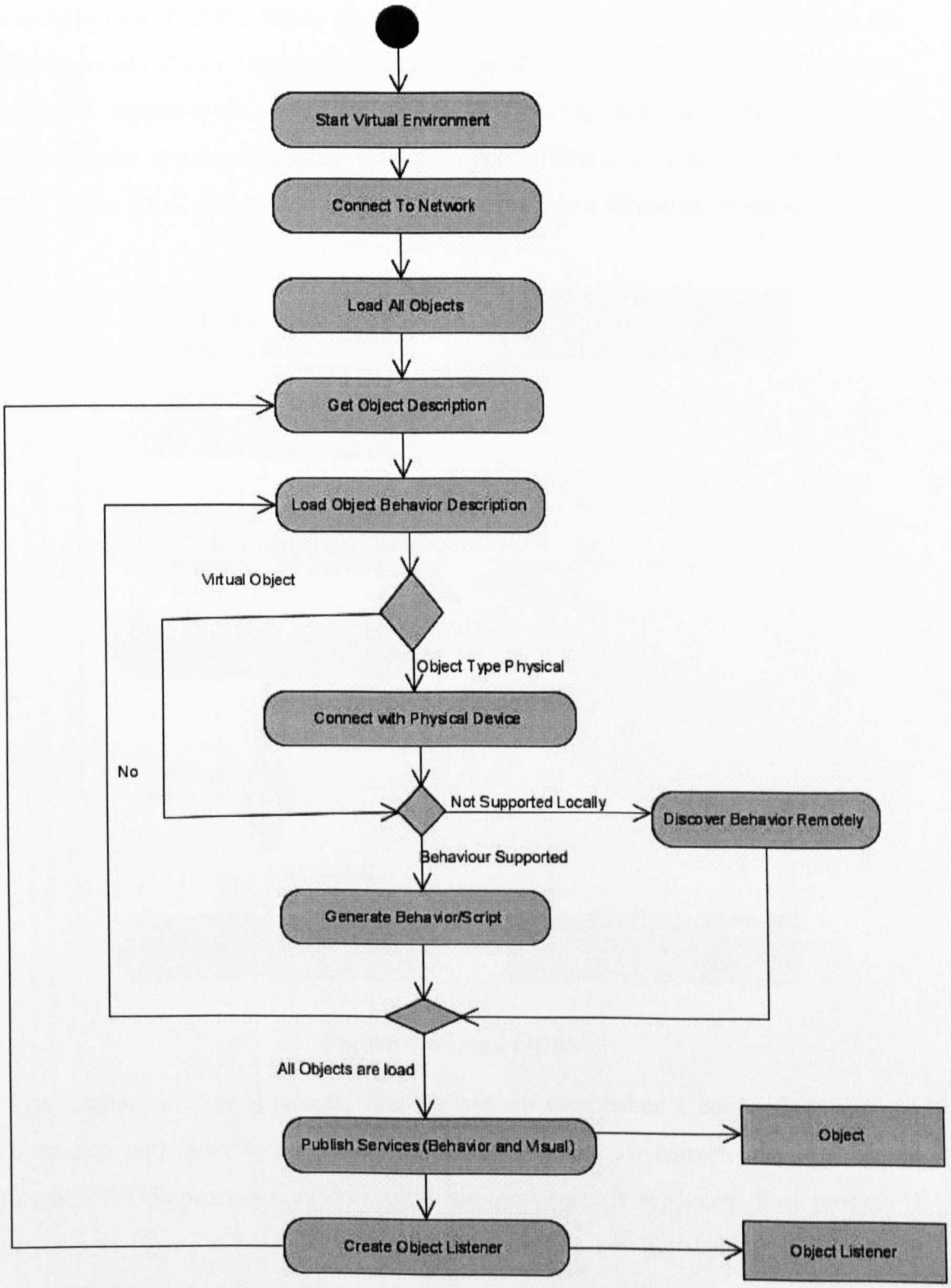


Figure 4.5 Start Virtual Environment

During the Start up procedure, the virtual environment connects to the P2P network and loads all objects from its data source. The whole environment is dynamic and nothing is hardcoded, all object functionality is created at run time using scripting

and a Rule-Base system. When an object is added to the virtual environment it's attributes are saved in a data source (i.e. the type of object it is, what type of scripting language it supports, the graphical format it supports, and the using the object description, the system determine what the type of object it is such as phsycial or virtual. Figure 4.6 illustrates how objects are loaded into a virtual environment.

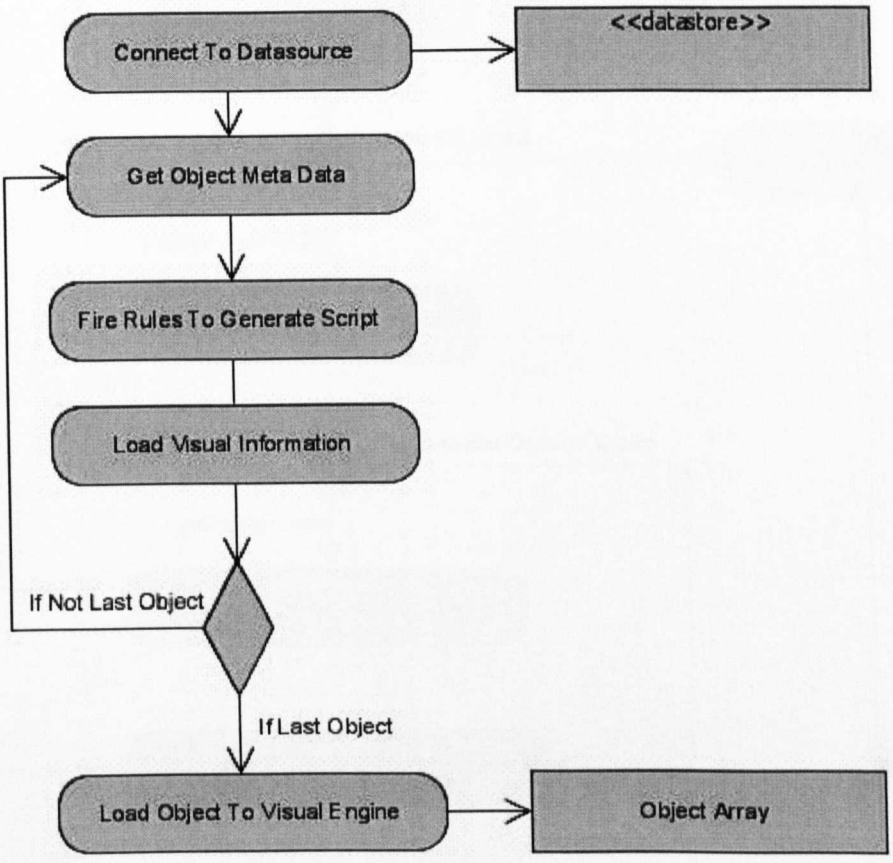


Figure 4.6 Load Objects

The loading process is simple, first the system establishes a connection with the data source and then loads object information such as behavioural and visual information. This process continues until the last object is retrieved. This process is illustrated in Figure 4.7 where it divides datasets into sub-set datasets if the object forms part of a wider composition.

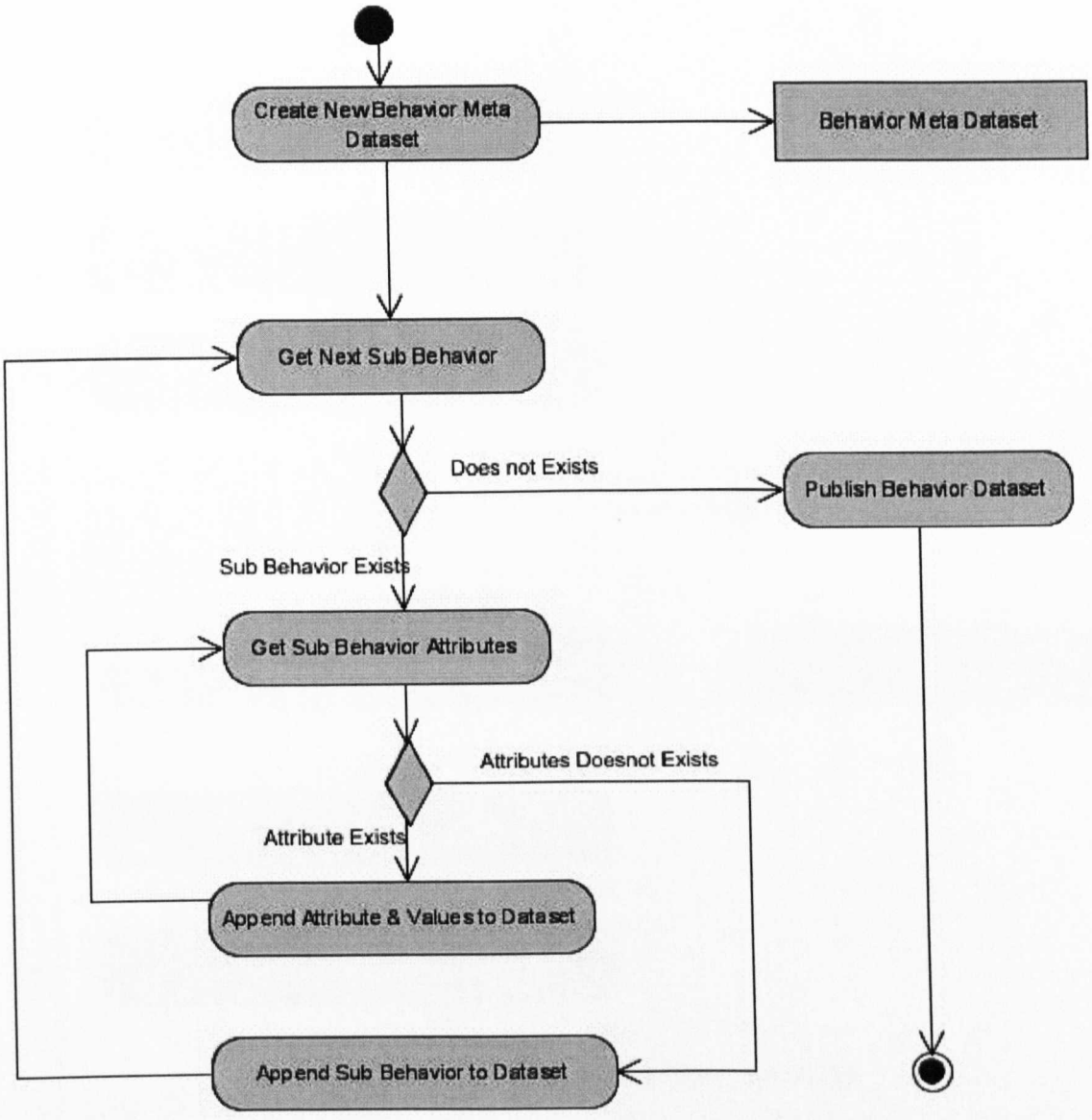


Figure 4.7 Load and Publish object Behaviour

Once an object has been created both its behavioural and visual models are registered with the target environment and saved to a physical location. The system then adds a unique advertisement description to that object and publishes the advertisement in the network as illustrated in Figure 4.8. This allows other peer users to search for objects.

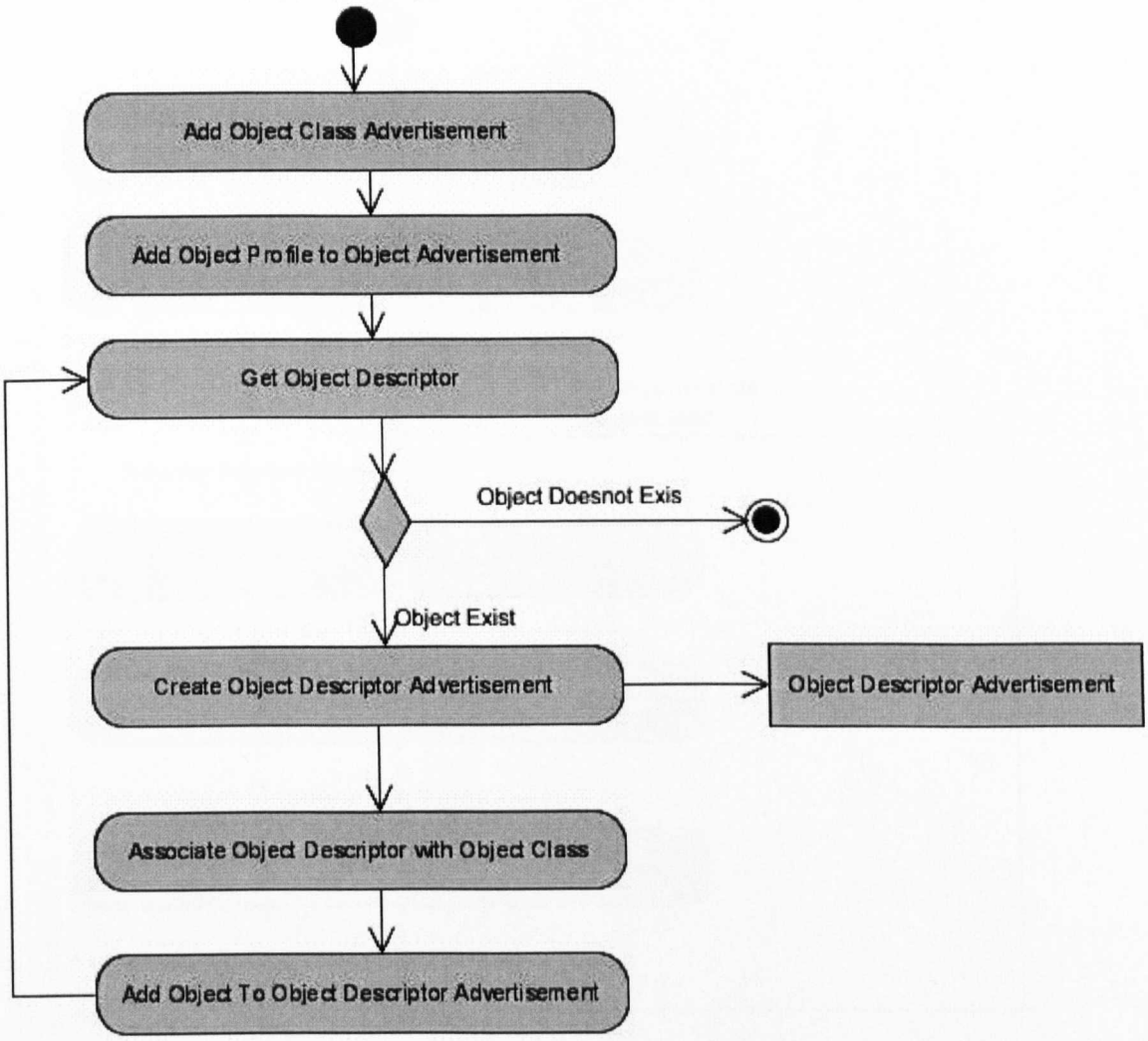


Figure 4.8 Object Behaviour's Advertisement

Behaviour services are special services where a peer only advertises behaviour translation services. For Example, the virtual environment can translate behaviours not supported by the destination virtual environment using translation services as illustrated in Figure 4.9. The system is designed to share objects between different virtual environments and it is difficult to generate behaviour on the target environment if the target environment uses different semantic descriptions. We have developed special services to translate behaviour descriptions between two different environments in order to generate a correct behaviour for the object.

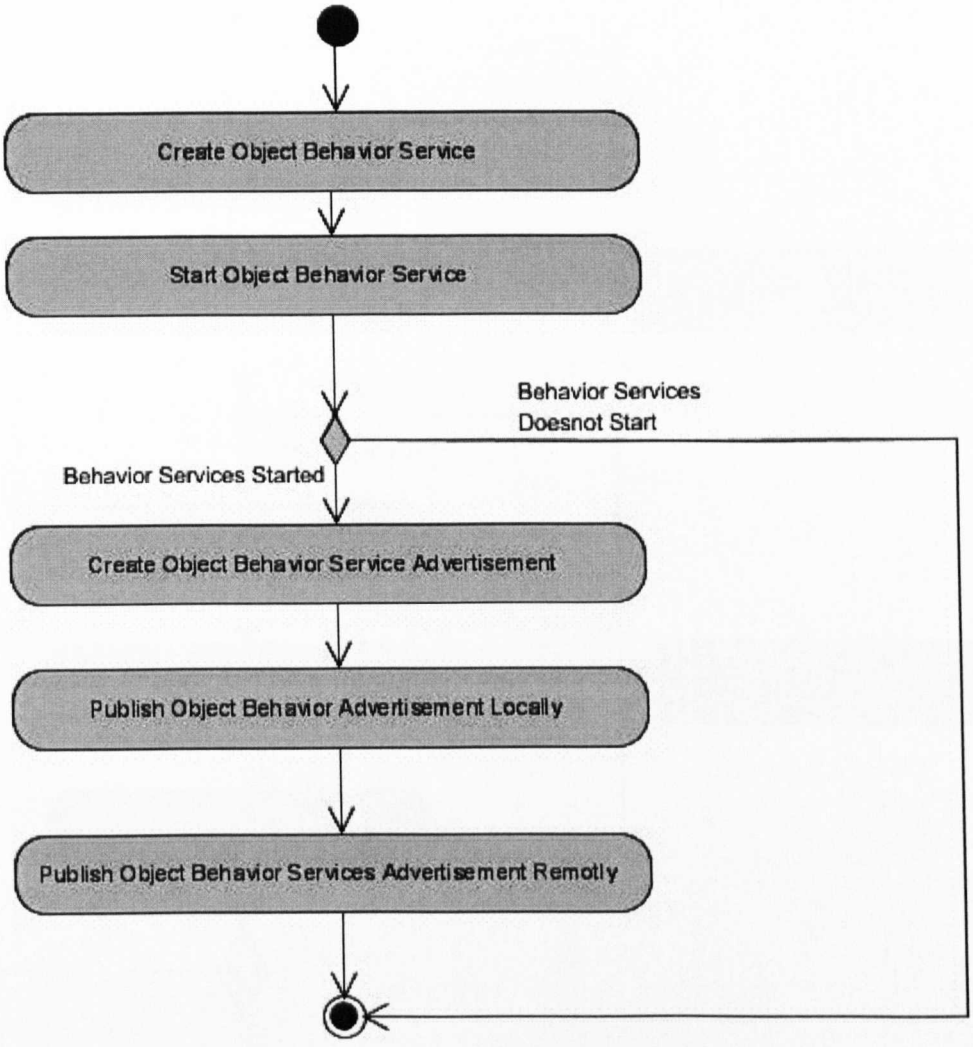


Figure 4.9 Behaviour Advertisement Services

Dynamically creating behaviours involves taking the semantic description of behaviours and inserting them into the working memory of a rule engine. Depending on the facts about the behaviours inserted and the behaviours supported by the target environment, scripted behaviours are dynamically created and assigned to the object before it is fully inserted into the target environment and used. Figure 4.10 illustrates how rules are fired against facts to generate behaviour dynamically. Firing each rule results in a script that has been generated and passed to the scripting parser module for validation.

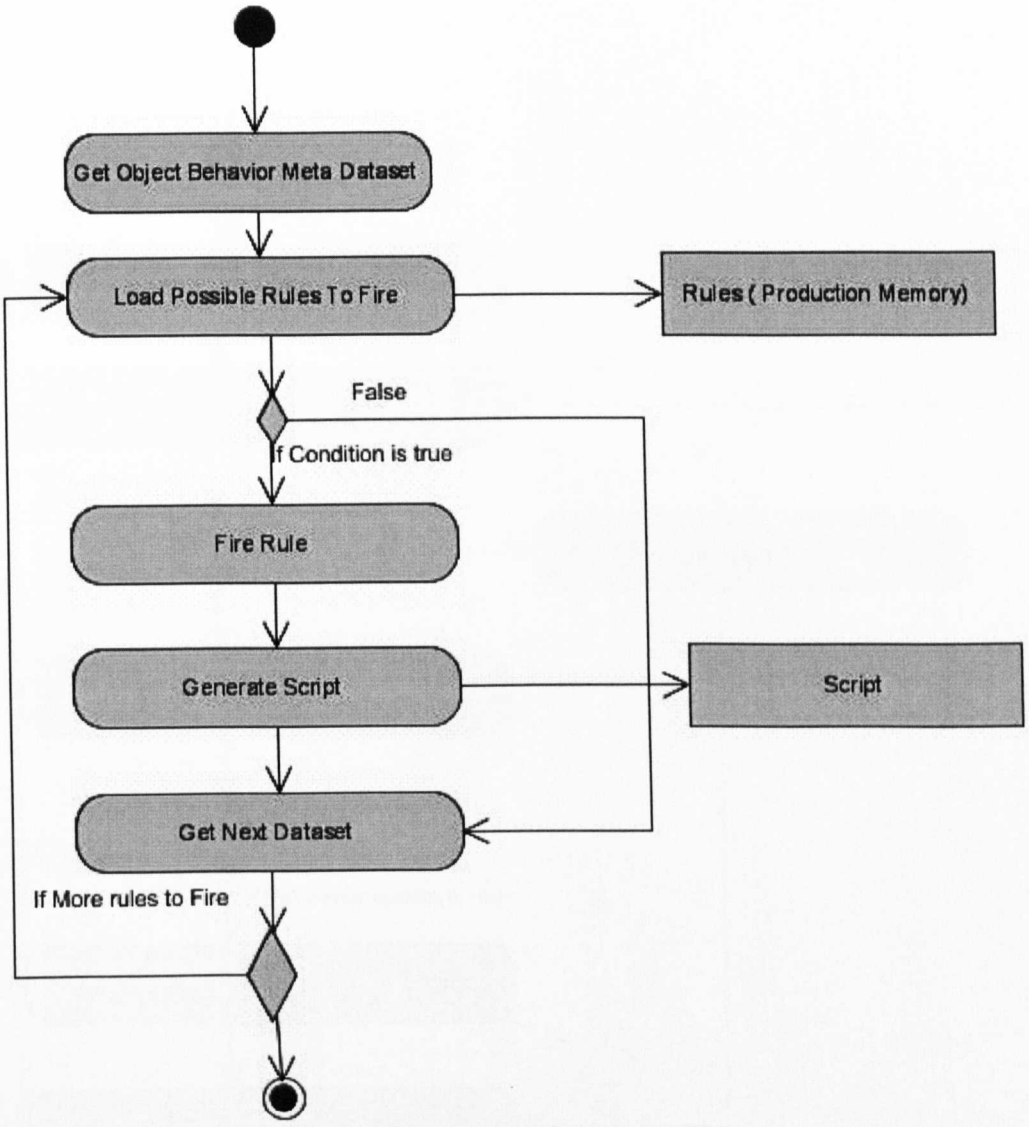


Figure 4.10 Generate Script using Rule Engine

When a virtual environment starts for the first time, it loads all objects. For each loaded object, information is used at run time to generate behaviours or load already generated behaviours from the cache as illustrated in Figure 4.11. Due to the dynamic nature of the system all objects are saved instead of being hard-coded into the program thus each time the system starts up it loads each object one by one and including its associated information and its status when it was last shutdown. The system also generates behaviours if they have not been generated previously by firing rules. If a behaviour cannot be generated it is discovered within the network. Once a behaviour is found the behaviour is registered, including it's behaviour, with the system.

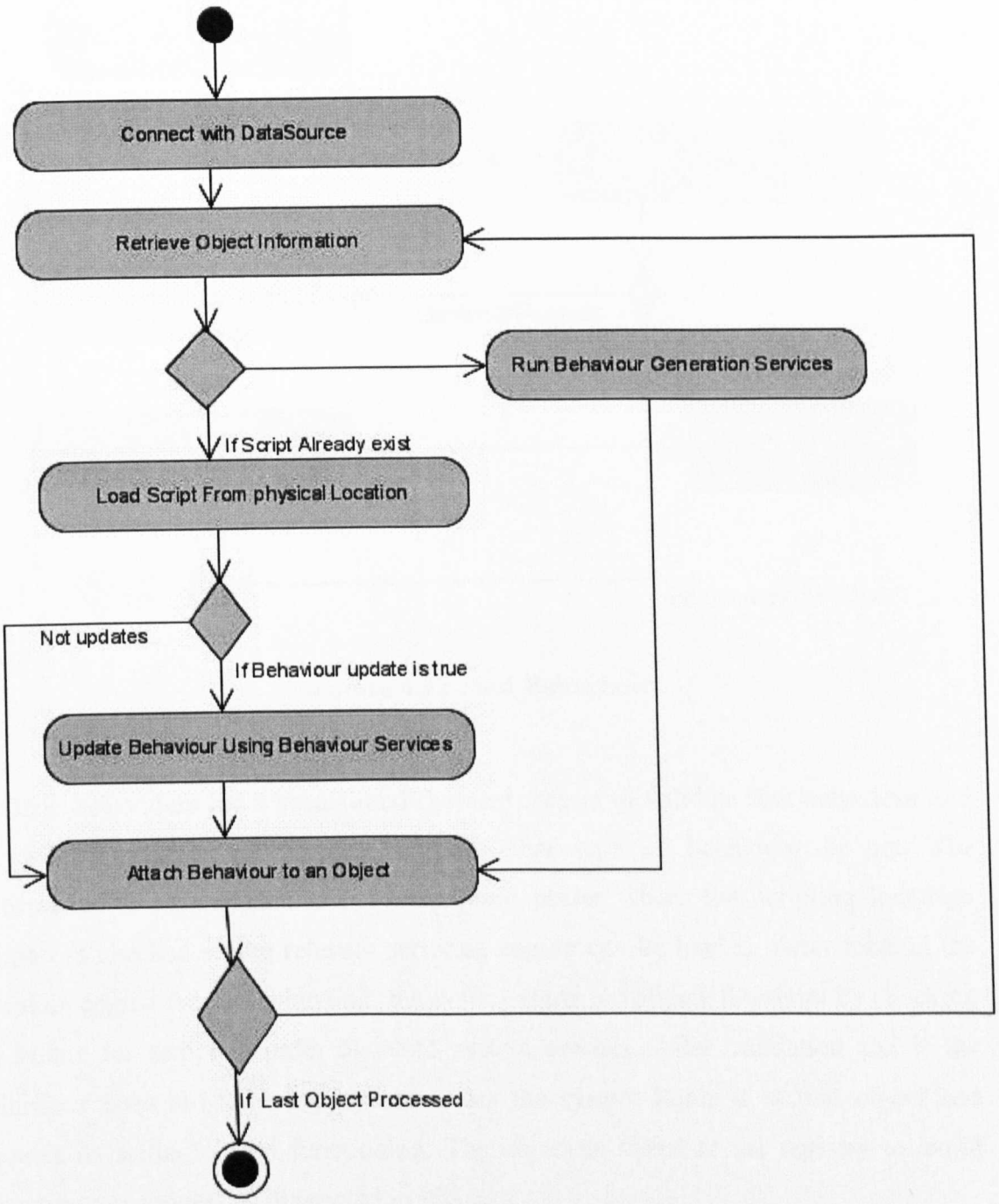


Figure 4.11 Load Behaviours from Datasource

Figure 4.12 shows the process for finding behaviours either locally or remotely. To find a behaviour it first checks locally. If an exact or similar behaviour is found locally then it evaluates the syntax of the script before assigning that script to the object. If the script syntax compiles it is assigned to the object.

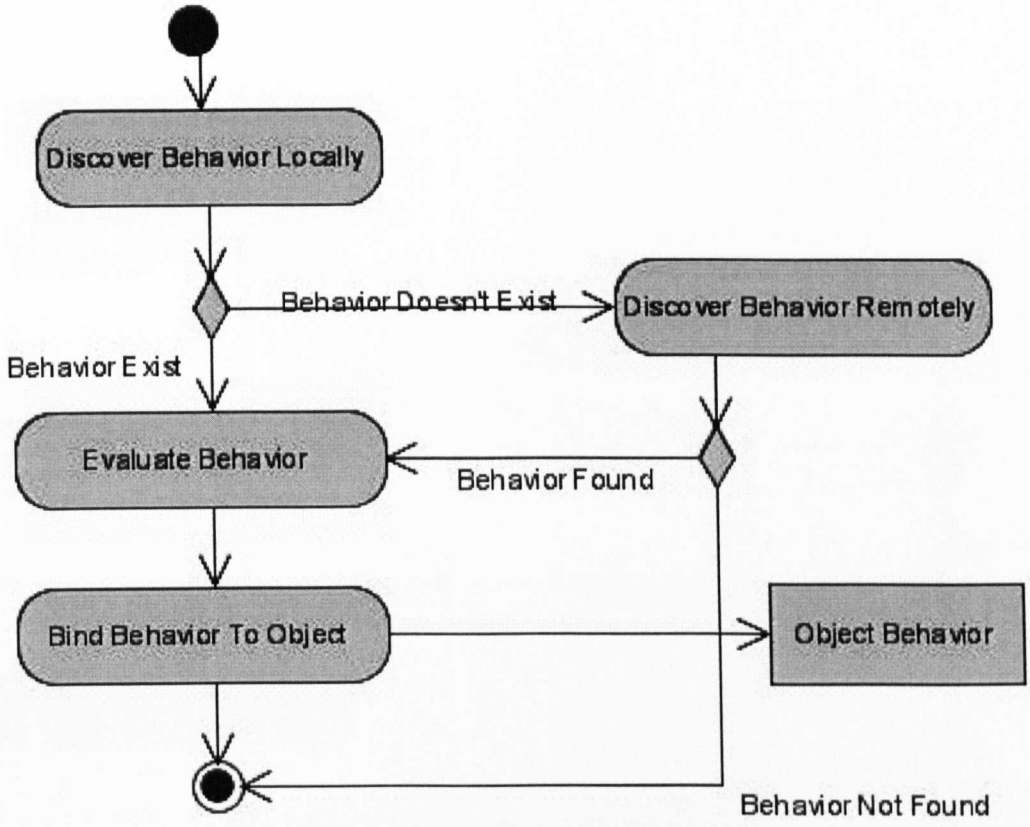


Figure 4.12 Find Behaviour

Once behaviours have been found the next step is to validate that behaviour and check if the target environment supports that type of behaviour or not. The information is then loaded into the scripting parser where the scripting language support is checked so the relevant scripting engine can be loaded. After loading the scripting engine for that behaviour, the system starts to validate the script by checking its syntax for errors in order to avoid system crashes. After validation and if the behaviour does not have any syntax errors the system binds it to that object and updates its status to start functioning. The object is saved to the registry to avoid repeating this process as illustrated in Figure 4.13.

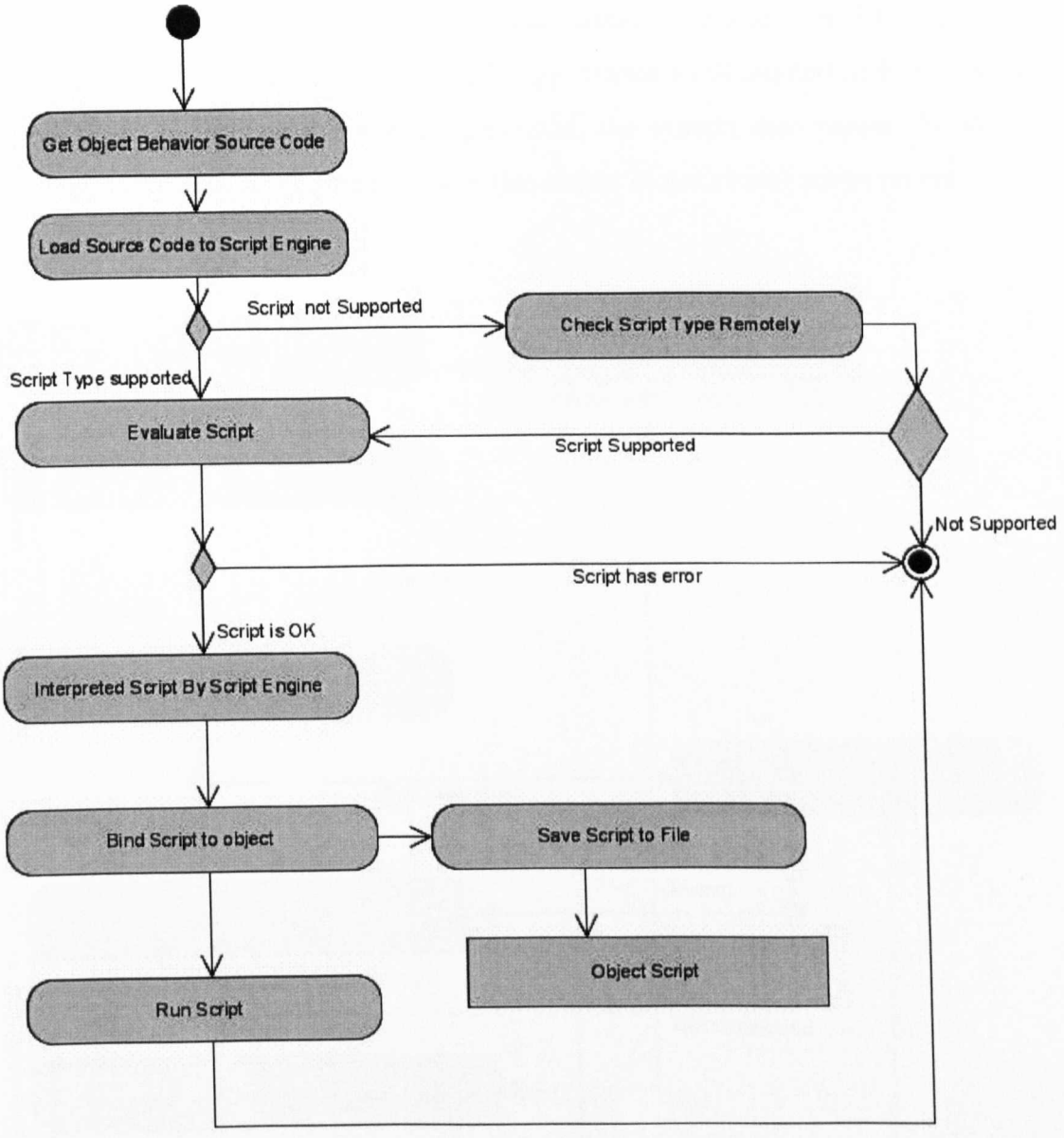


Figure 4.13 Script Validation

When an asset is passed to the Visual Resource Manager, it is registered with the resource Assets/Lookup. Following this it extracts the meta-data for that object and passes it to the Visualization Engine which in turn renders the 3D object into a graphical shape. At the same time, the Behaviour Matcher looks up the behaviour of similar objects in that environment. If it finds the behaviour then it assigns it to the object along with the effects it supports when it is executed. Let us assume that a user sends a game object to another game. The game object's behaviours should also be transferred from the source environment to the destination environment so that the user can fully enjoy the new object features, such as its graphical special effects or how it reacts to stimulus from the game, such as being shot at. A mapping is

performed where meta data is extracted for the visualisation and passed to the visual engine, which it used to render the object's appearance as illustrated in Figure 4.14. Once all object behaviours have been generated, the system then passes the visual information to the Visual Engine to render that object in the virtual environment.

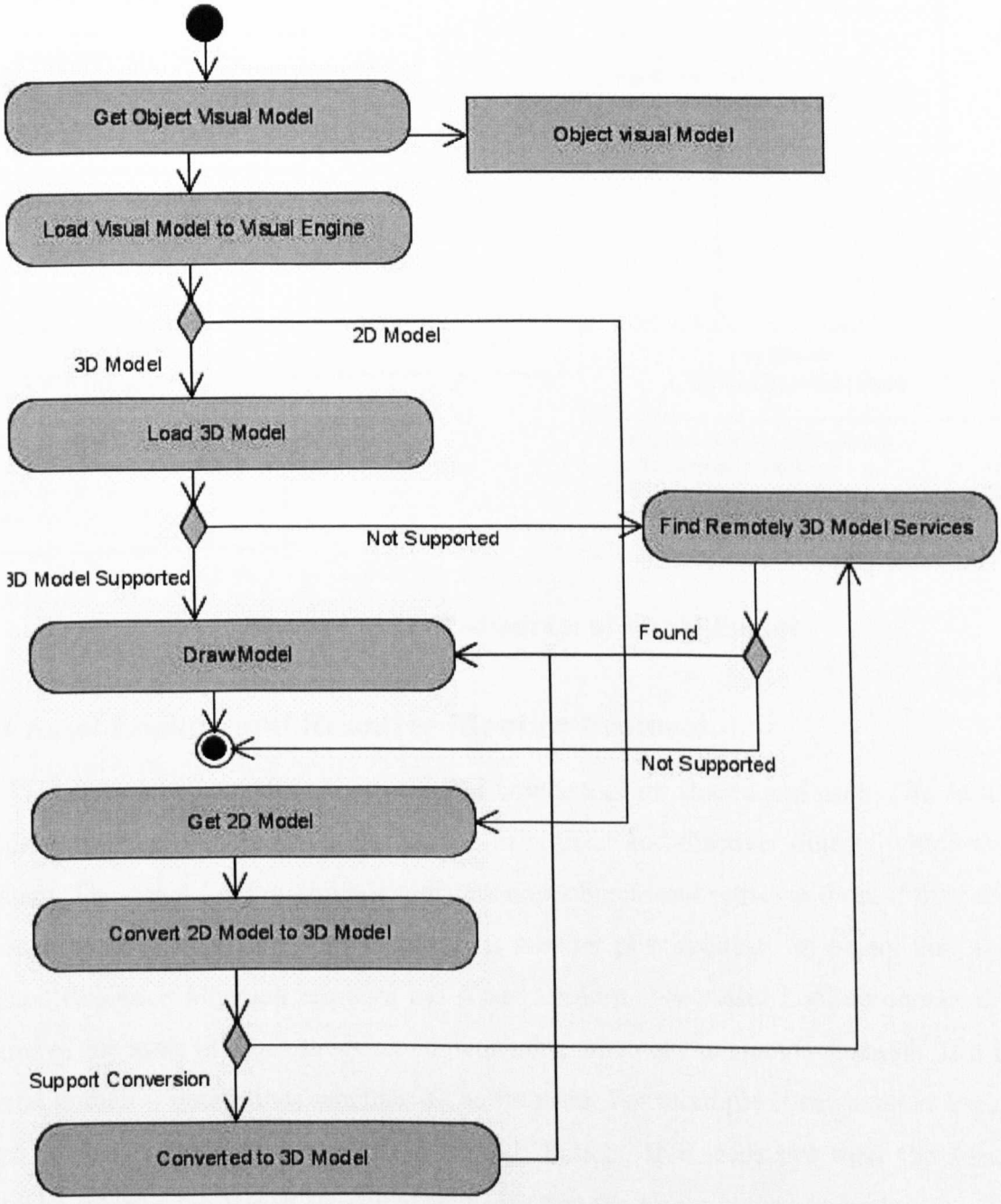


Figure 4.14 Visual Engine

The system checks either its 3D model or 2D model to draw and look for services to convert the 2D model to 3D and if it can successfully translate from 2D to 3D then

it will draw that object in the virtual environment. The class diagram shown in Figure 4.15 shows how the system displays and renders objects in the virtual environment.

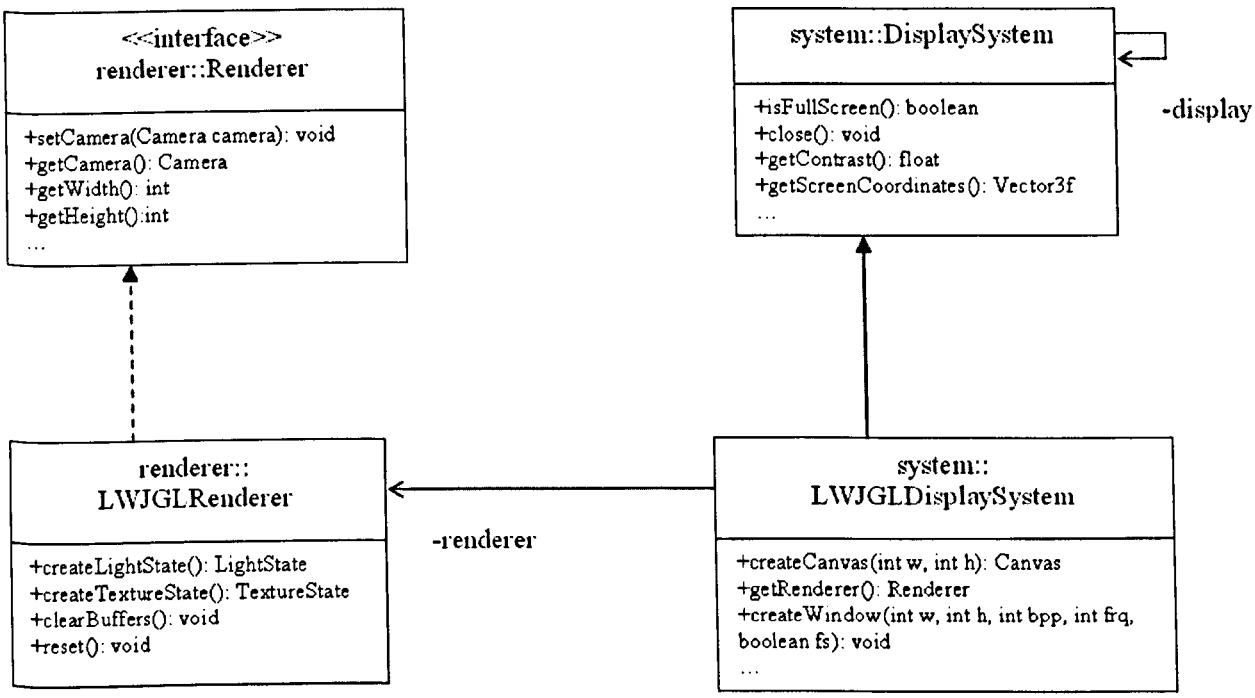


Figure 4.15 UML-diagram of Visual Engine

4.4 Asset Lookup and Resource Monitor Protocol

This section begins with a discussion of how assets are shared and used. The Asset Lookup module implements a mechanism to register and discover objects within the system. The Asset Lookup module registers new objects and retrieves them if they are needed by another system. For example, if another peer requests an object then the Visual Resource Manager requests the Asset Lookup. The Asset Lookup checks the status of the asset in the data source to determine whether the asset is sharable. If it is sharable then it determines whether it can be used. For example if the asset is a gun then it checks whether the gun has enough bullets, if it does not then the Asset Lookup informs the Visual Resource Manager that the object is not available.

4.4.1 Protocol requirements

In heterogeneous environments, (both real and virtual) monitoring and sharing resources with another environment can be challenging. Key requirements to achieve this are as follows:

- The Protocol must provide services to update information associated with objects.
- The protocol must provide services to associate object management capabilities on a per-object basis.
- Some managed objects represent shared resources that might be referenced by multiple objects. The protocol must provide services that allow an object to associate an existing shared resource object with other registered objects.

4.4.2 Protocol Operation

The Asset Lookup and Resource Monitor register objects in the main repository with associated values and keep track of the object resources used. As discussed in section 4.4.1, the protocol uses an object management system to monitor the level of usage. For example, if you add a sunSpot as an object then it is likely the sunSpot uses battery power.. The level of sunSpot battery usage can be determined by how much battery powered has been consumed. The data structure is illustrated in Figure 4.16.

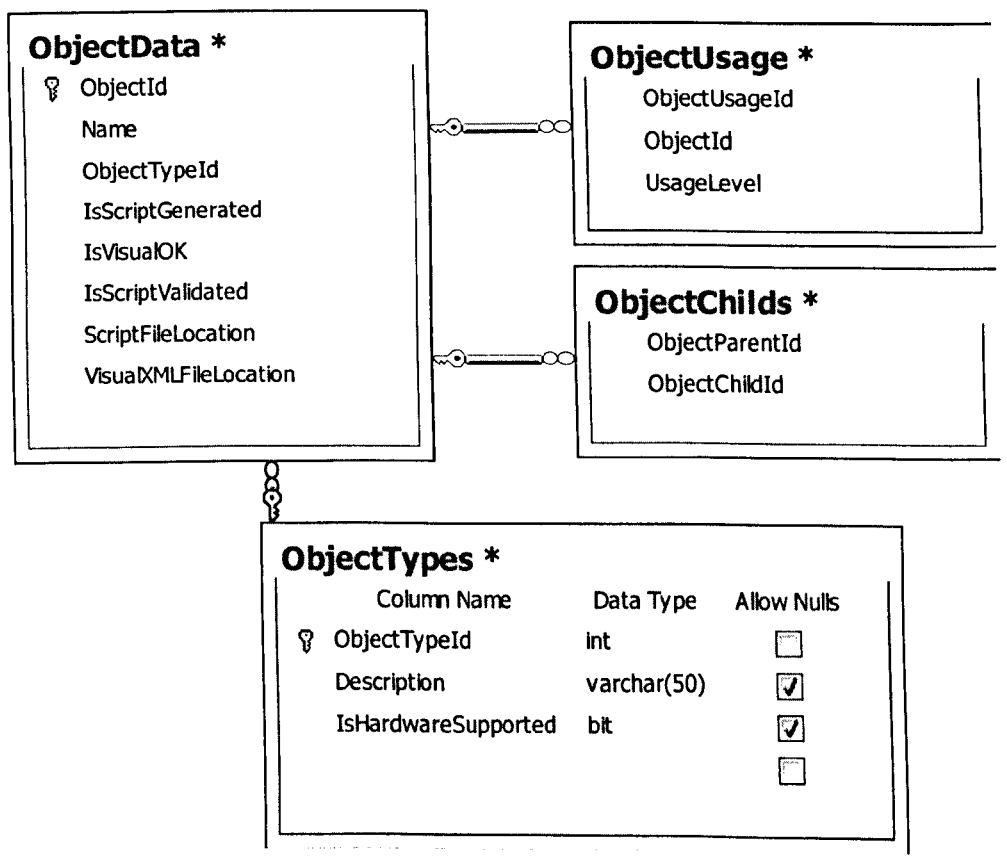


Figure 4.16 Data Model

When an object has been stored in the data source the system generates a unique ID to identify it and its associated child objects as show as in Figure 4.16, including the level of usage it has. The process is illustrated in Figure 4.17.

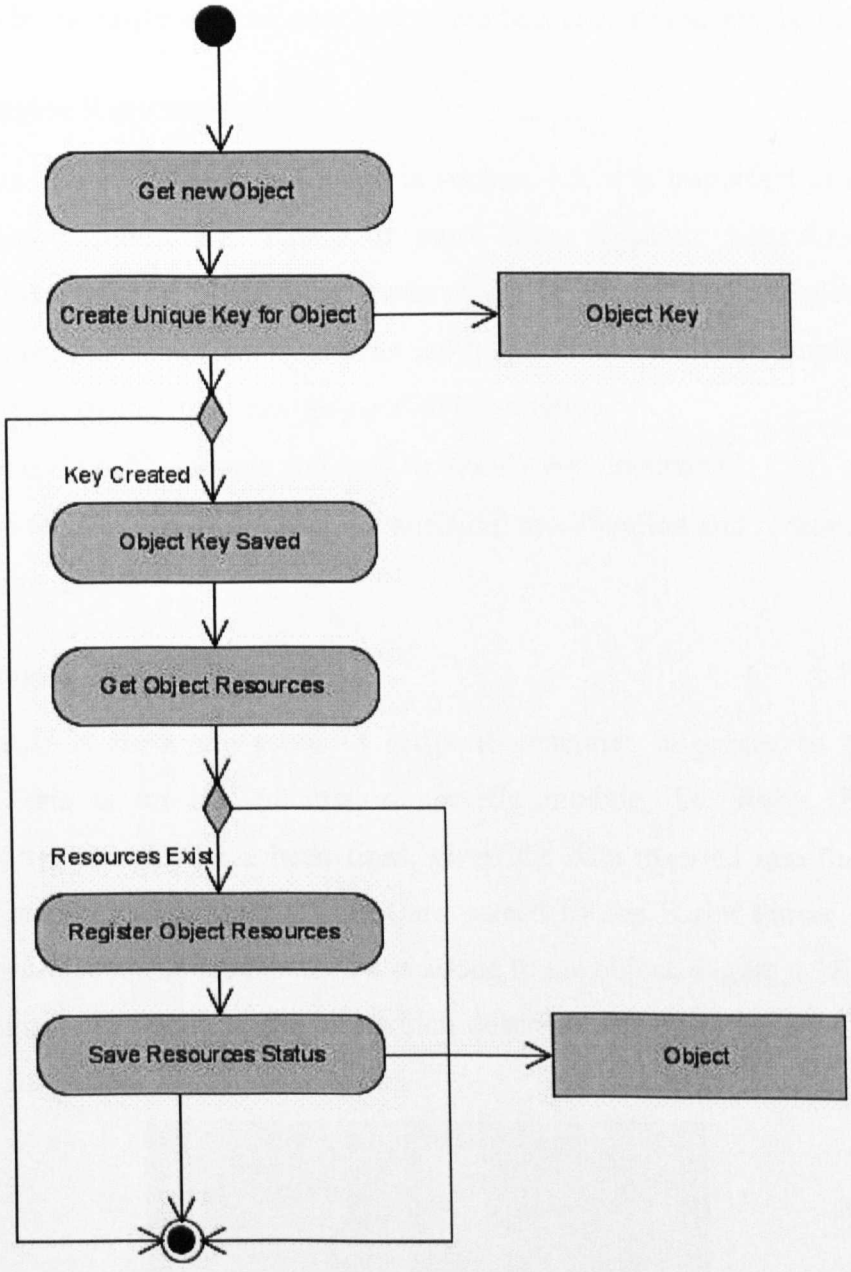


Figure 4.17 Assert Lookup/Resource Monitor

4.5 Rule Engine

The Rule-Based system provides a powerful way to dynamically script behaviours from semantic descriptions [Merabti 2008b]. In this section, we build on these ideas to dynamically create object behaviours using Rules.

Rules are run against known facts about the behaviour objects support and dependent on matches, relevant rules are fired – these are also referred to as Production Rules. Within the actions of fired rules script functionality is constructed that is known by the target environment and which best accommodates the behaviour.

4.5.1 Rule Engine Requirements

As we have discussed the Rule Engine in section 4.5, it is important to address a number of key requirements. Firstly, it must allow dynamic behaviours to be generated. This can potentially take many rules to create the complete set of behaviours in heterogeneous environments (both real and virtual). To implement the rule engine the following requirements need to be satisfied.

- Rules should be simple and easy to modify and understand.
- The System should support the addition; modification and removal of rules at run time.

4.5.2 Rule Engine Design Overview

As each rule is fired, the piece of script it generates is passed to the Script Generator – this is an implementation specific module, i.e. Ruby, Python or JavaScript. After all rules have been fired, given the data inserted into the working memory, the combined sections of script are parsed by the Script Parser where the syntax and structure is validated before it is added to the object. Figure 4.18 illustrates the class diagram for the Rule Engine, which describes the class variables used and the methods supported.

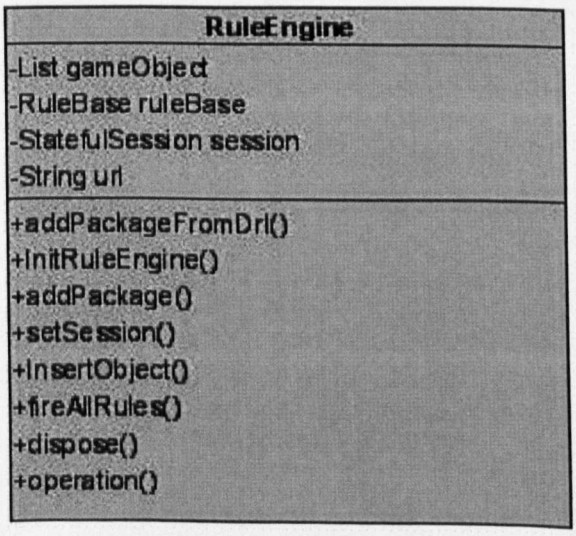


Figure 4.18 Rule Engine Class Diagram

This is not to be confused with the object being processed but the object being created for the target environment. This object contains the visualisation information and the scripts required to project the object into the target environment. Constructing the scripted behaviour is an iterative process, which could result in many behaviours being added to the object. For example, a lamp could have a TurnOn, TurnOff, and Flash behaviour. This would result in three scripts being generated.

Figure 4.19 illustrates how Data is inserted into the Rule system and how the system determines which Rules should fire to generate the behaviours. Once the right set of rules has been selected, the system fires each of them and this results in a script being generated.

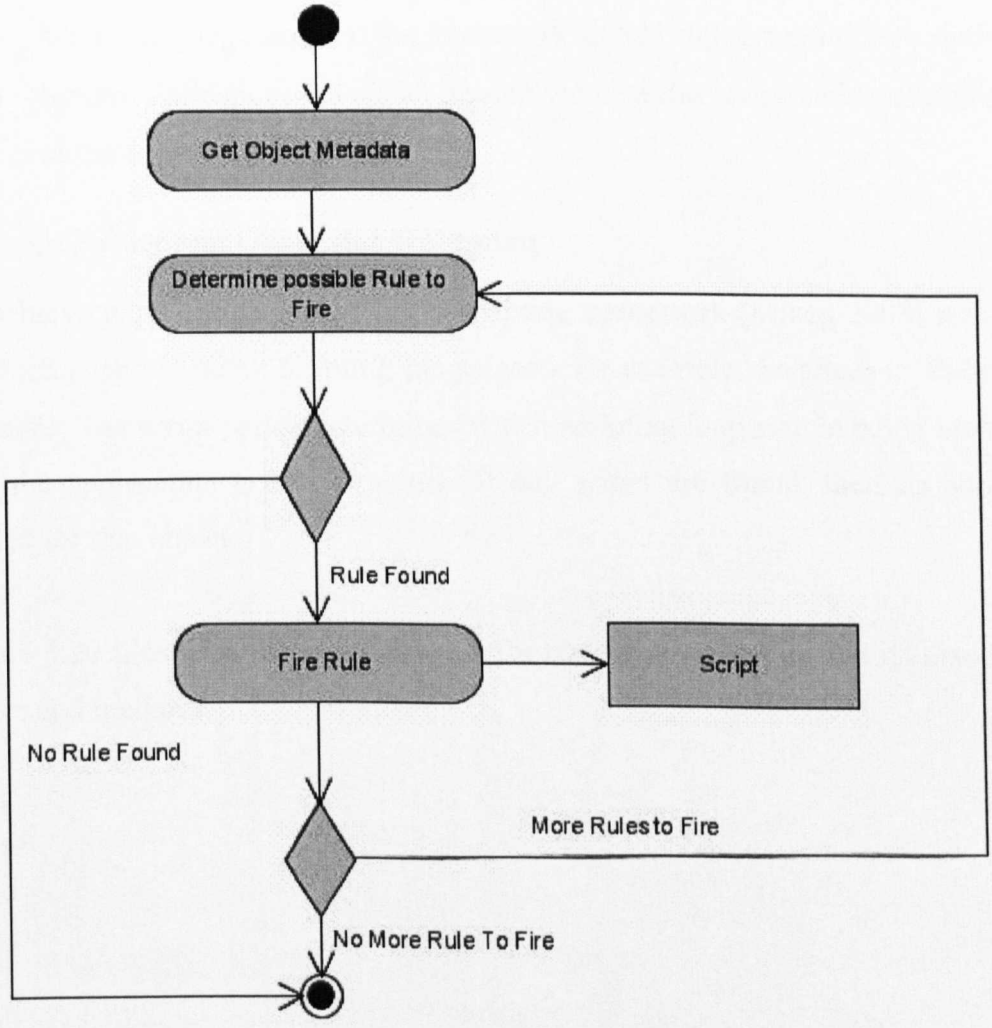


Figure 4.19 Rule Process

4.6 Script Parser and generator

Each rule fired generates a piece of script, which is passed to the Script Generator – this module generates the script according to the scripting language being used. Once all rules have been satisfied result many scripts have been generated, the combined sections of script are parsed by the Script Parser where the syntax and structure is validated before it is added to the object.

4.6.1 Script Parser and Generator Requirements

To generate scripts and assign that behaviour to an object several requirements need to be satisfied.

- Multi Language support: the framework should support multiple scripting.
- Syntax Validation: the parser should validate the script before assigning it to the object.

4.6.2 Script Parser and Generator Operation

To achieve multi language support, a scripting framework [Mikeg 2008] has been used which supports many scripting languages - for example, JavaScript, jRuby and many more. The script parser determines which scripting language is being used and checks the appropriate syntax structure. If any errors are found, then no script is generated for that object.

Figure 4.20 illustrates the class diagram for the scripting engine and its associated variables and methods.

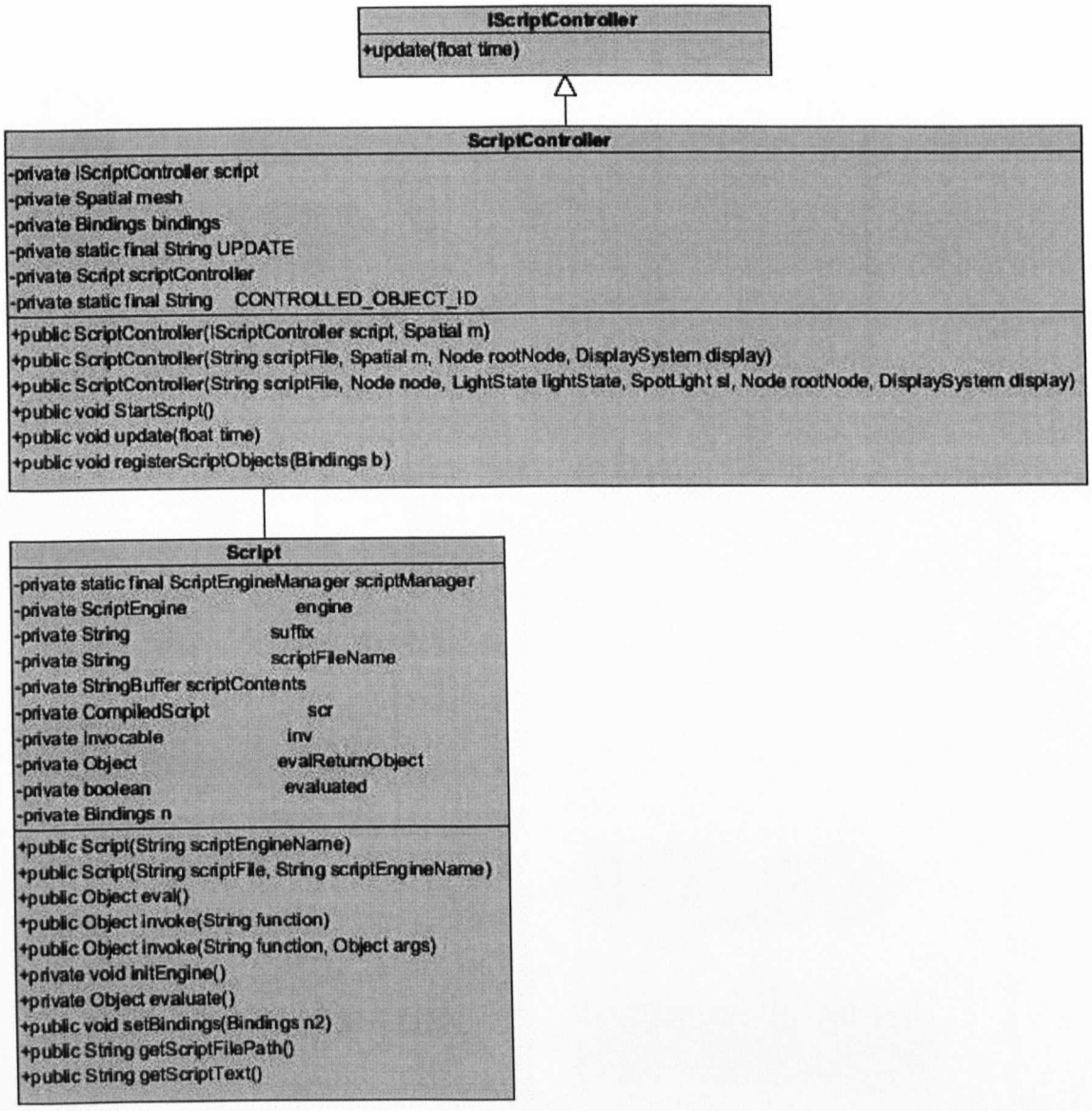


Figure 4.20 Script Engine Class Diagram

It is more complicated to validate script syntax where the system has to determine the right scripting engine to validate that script. The main problem occurs when the script is correct, but system load the incorrect scripting engine to validate the script. To avoid this problem the system transfers information to the source environment about the script language type. In order to validate scripts the system first loads the appropriate scripting engine. If the system does not find the right scripting engine for the source script in the target environment it notifies the user that objects cannot be loaded and saved – information contained in the system is used to discover it later as illustrated in Figure 4.21.

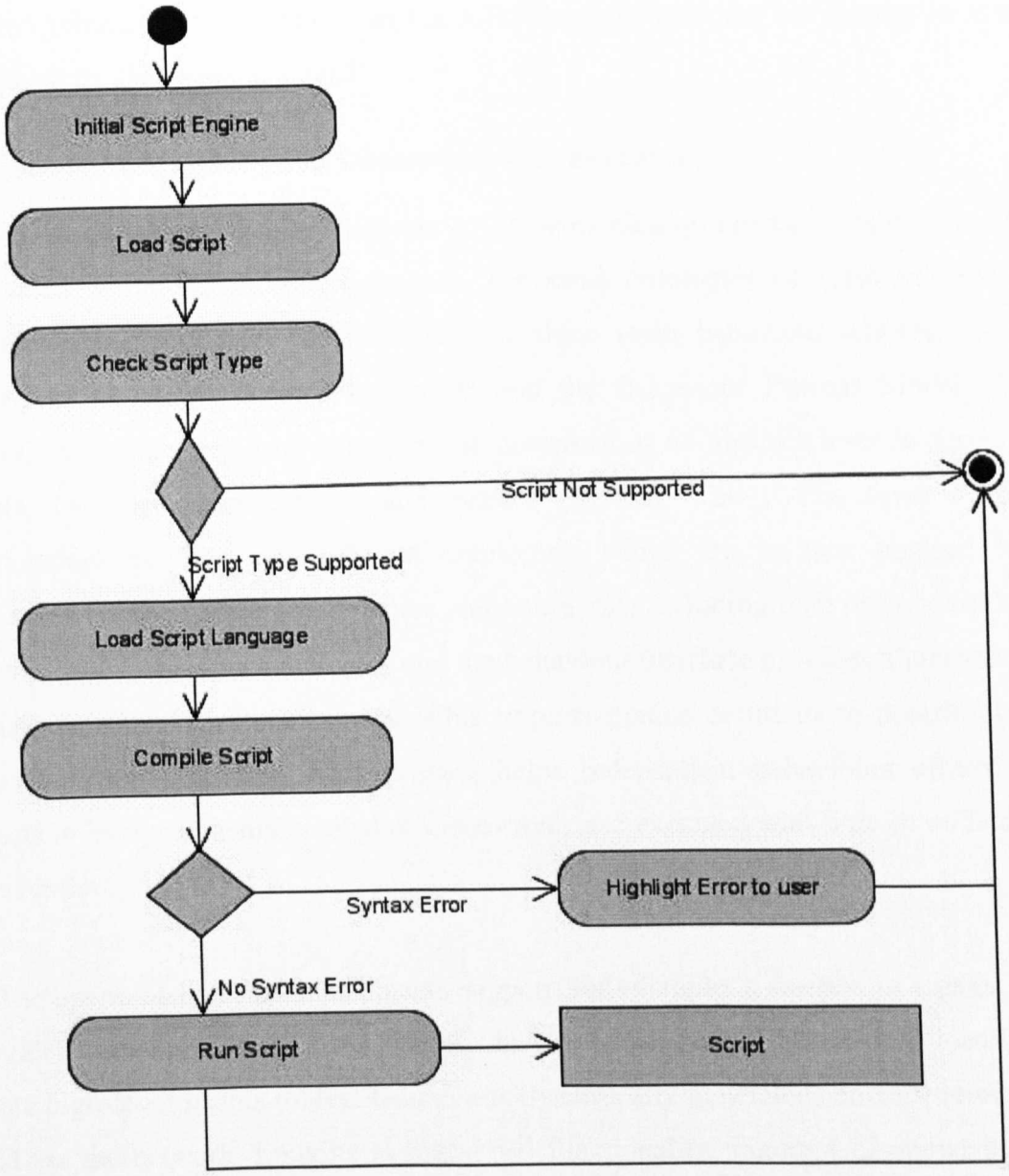


Figure 4.21 Scripting Engine

4.7 Behaviours Ontologies

As discussed above shared assets are described using two layers, the first is the visualisation and the second is the scripted behaviours. A mapping is performed between the object and the game engine by extracting the Meta data used to describe the object and its 3D characteristics. The rewriting scripting engine is used to find appropriate behaviours the game engine can accommodate; this is detailed in the

scripted behaviour of the object. In the following sub-sections, we discuss in more detail how this has been achieved.

4.7.1 Semantic Matching and Generation of Behaviours

Behaviour ontologies allow objects to be semantically annotated, which can be embedded within object advertisements. The same ontologies can also be used to describe object requests. We currently use three main behaviour ontologies; the Behaviour Ontology; Behaviour Profile; and the Behaviour Process Model [48]. Behaviour ontologies allow objects to be described at an abstract level in terms of Inputs, Outputs, Preconditions, and Effects (IOPEs). The IOPEs form explicit relationships between the different ontologies, which are in turn mapped into signatures (method names, parameters and return data including type information). In this way, the Behaviour ontologies and the behaviour interface provides a mechanism to link semantic descriptions to possible implementation solutions to describe how behaviours are generated. This process helps independent behaviours offered by objects to be dynamically created or discovered, and executed with little or no human intervention.

The operational capabilities objects support, for example, a weapon in a game can provide functions to reduce the life of objects, which can be either used locally to create high-level scripts (object behaviours dynamically generated) or discovered and used via the network. Looking at high-level functionality, Figure 4.22 shows how a weapon can be created in an environment (an object behaviour that does not exist but rather emerges through the simulation of object behaviours) where assumptions could be made between being hit by a vehicle and being shot by a weapon.

Using the concept of IOPEs, object behaviours can be dynamically generated by matching similarities between the object request and the behaviour ontologies. Behaviour ontologies, in conjunction with domain ontologies used by the target environment the object is being projected into, are matched through a one to one mapping or semantically if vocabularies are syntactically different but semantically equivalent.

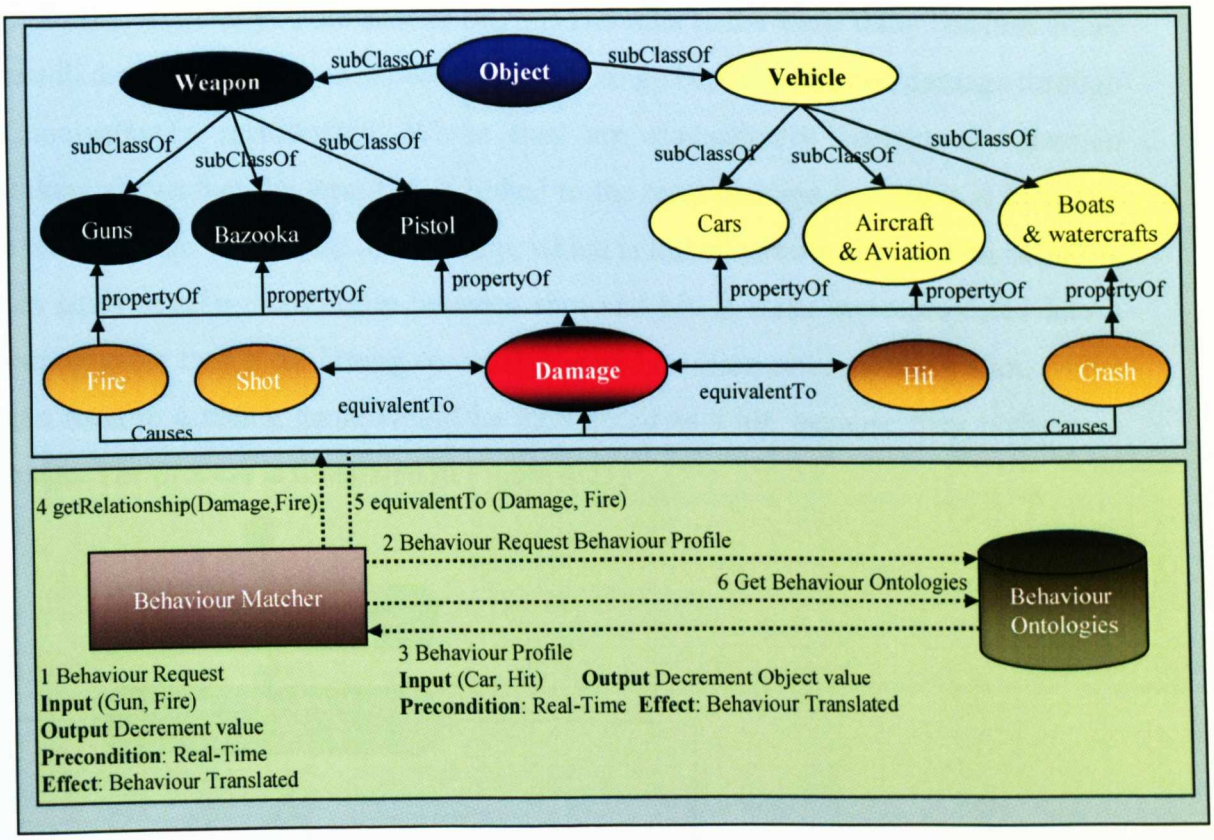


Figure 4.22 IOPE Matching and dynamic generation of behaviour

Looking at Figure 4.22, the user submits an object request to the target environment (step 1). Upon receiving the request, the behaviour matcher in the target environment begins by iterating through the Behaviour Profiles for each behaviour it provides (step 2). Using the domain ontology, the IOPEs in the Behaviour Request are matched with IOPEs in the Behaviour Profile. If exact matches are found then the process simply moves onto the next IOPE. However, if there are syntactic differences, the two terms are passed to the domain ontology [Merabti 2008a] (step 3 and 4). If a relationship exists between the two terms, a match has been found that semantically links the two terms together.

For example, if we were trying to project a weapon, such as a gun, into an environment typically designed to have lots of cars then we would have to make an interpretation about what fire means and the effect it has on objects in that environment. For example, looking at Figure 4.22, we could use the domain ontology to work out this effect through the semantic links, i.e. firing at something causes damage much like a car does when it crashes into something. The object subjected to being shot at or hit results in being damaged. Consequently, we can use the ontology

to infer that whether you are shot or hit, this amounts to the same thing because either one will damage you. Step 5 shows a simple linkage between fire and damage through an equivalentTo relationship. Whilst they are syntactically distinct, the domain ontology shows that the term fire is linked to the term damage and crash is linked to the term damage via a cause relationship, which is linked to both shot and hit resulting in an equivalentTo relationship between shot and hit. If a car were projected into a shooting game then a car hitting something would be interpreted as being shot, whilst a gun fired in a racing game would be interpreted as a hit, because they both cause damage. The process is illustrated in Figure 4.23.

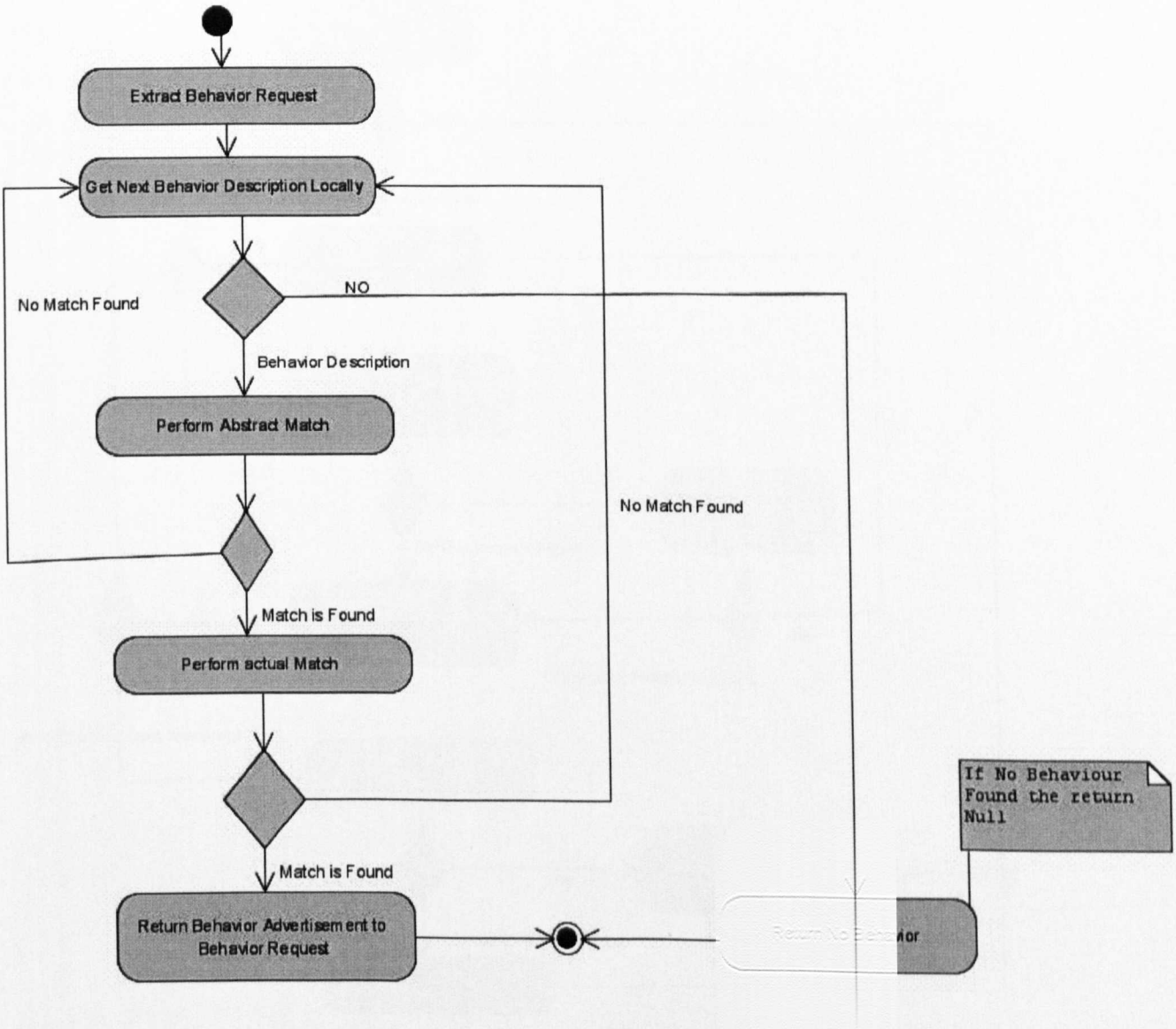


Figure 4.23 Process Behaviour Request

During the matching process, a table is created containing the matched IOPEs from the behaviour request. The matched IOPEs act as keys in the table and have corresponding values, which represent the names of the IOPEs used in the behaviour

ontologies. This process is important because the behaviour request and behaviour ontologies may refer to semantically equivalent IOPEs differently – the table of key-value pairs (stored in every environment capable of performing semantic interoperability) creates a semantic mapping between the different terms used. If all IOPEs in the behaviour request are matched this constitutes an abstract match and all ontologies associated with the Behaviour Profile (step 6), being retrieved as illustrated in Figure 4.24.

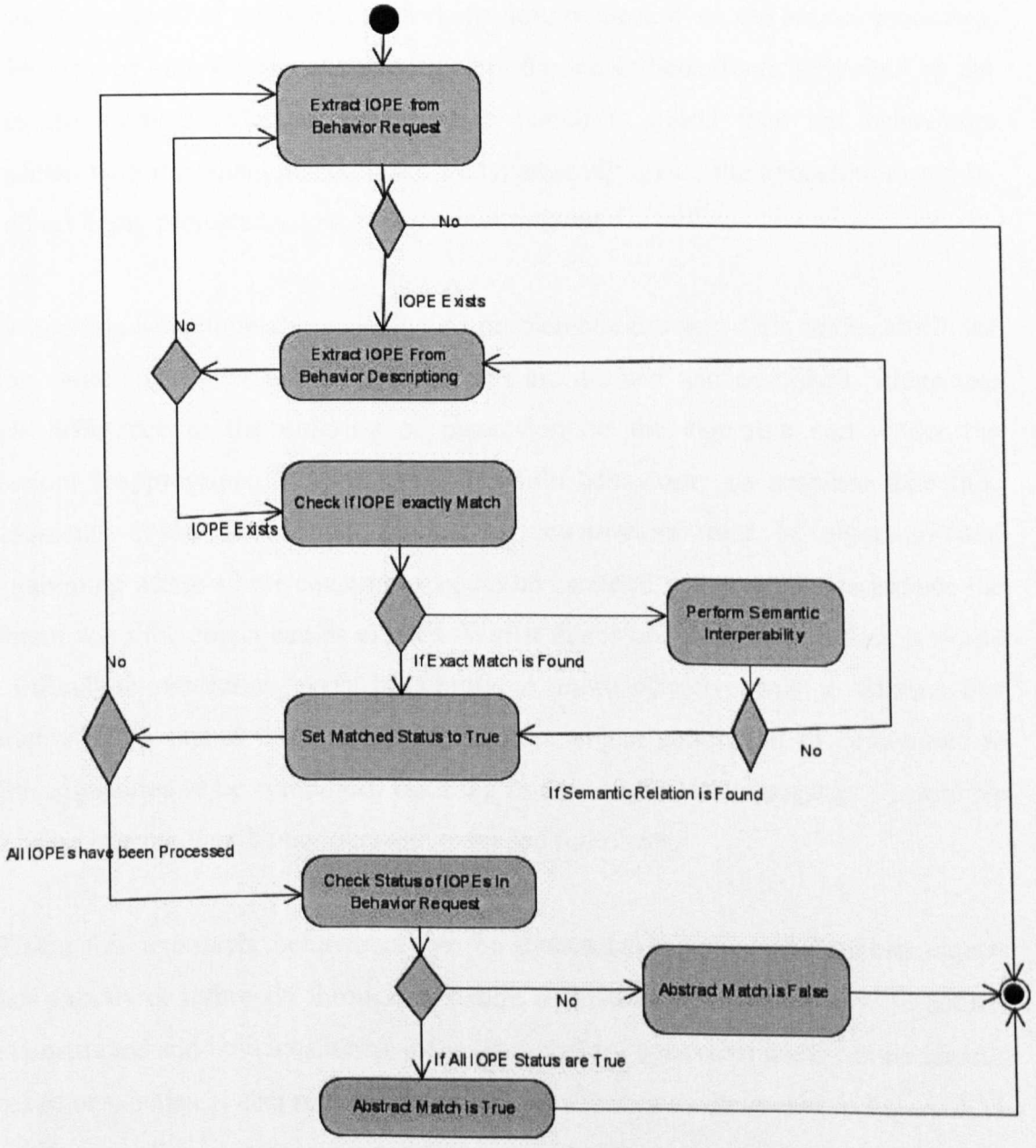


Figure 4.24 Perform Abstract Match

To determine whether the IOPEs defined in the behaviour request can be bound directly onto a signature provided by the target environment (a method with supporting input parameters and a return value), the Behaviour Profile is processed and the values associated with each IOPE are retrieved. These values specify which Atomic Process each IOPE belongs too in the Behaviour Process Model. The IOPEs may have been matched at an abstract level however, they could belong to different atomic processes. Therefore, the framework tries to determine if a single atomic process supports all of the IOPEs in the behaviour request. If so, the atomic processes, can be mapped onto the signatures defined in the list of behaviours supported by the target environment [Merabti 2008a]. If a match is found then the behaviours associated with the signatures are used to dynamically create the behaviours used by the object being projected into the target environment.

Generating behaviours dynamically is a problematic process. This can be attributed to the variation in how behaviour interfaces are defined and described, where one single difference in the ordering of parameters in the signature can render the behaviour inappropriate for generating dynamic behaviour. To accommodate this, mechanisms could be adopted similar to constructors used in object-oriented programming where a base constructor could be used and then extended to include the different ways the object can be created. Whilst this is one possible solution, it would be difficult to pre-define every possibility. A more effective way to address this limitation is to extend the concept of dynamic object generation of behaviours to enable signatures to be composed, resulting in new signatures emerging. We achieve this using intermediary behaviours and extended interfaces.

Using this approach, behaviours can be dynamically generated between objects either directly or indirectly through signature re-writing. It describes how signatures are constructed and indicates whether the intermediary behaviour itself can be directly invoked or whether it also requires intermediary services as illustrated in Figure 4.25. This process allows environments to dynamically discover and generate behaviour conflicts that may occur and proactively establish compositions with intermediary services. This may result in several candidate objects that provide the same functionality.

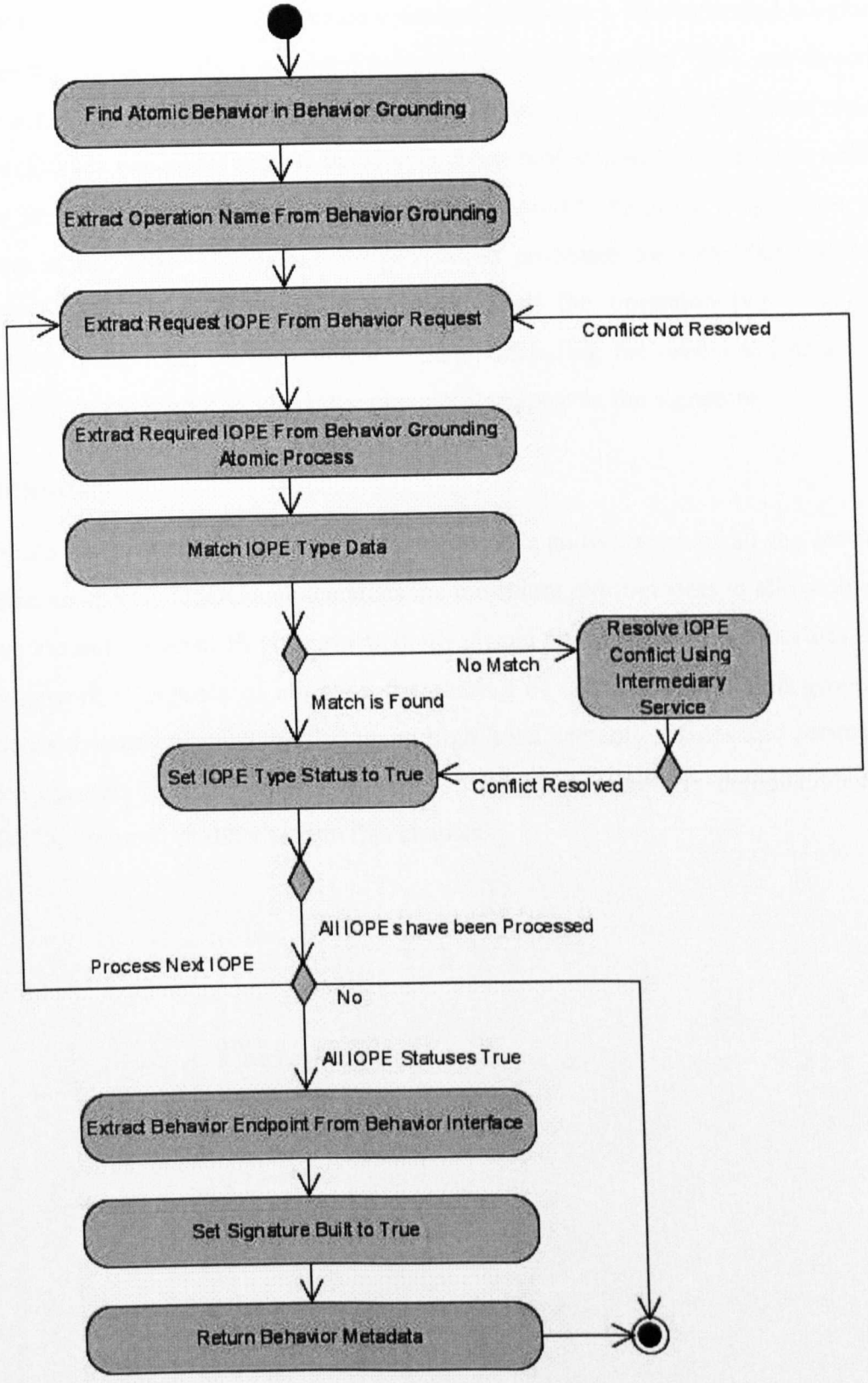


Figure 4.25 Build Signature

Behaviours that best match the object capability requirements defined in the object request are added to an extended interface metadata object (an XML file that represents the re-written signature. It may comprise references to any number of

different behaviours in order to produce a desired behaviour). The extended interface (EI) object is invoked when a behaviour provided by the object does not directly support a method invocation. This object behaviour has a fixed operation name called 'EI' which takes two parameters – the first is the extended interface metadata object and the second is an array containing all the parameters required to generate the behaviour at run time. This behaviour generation processes the extended interface metadata object, which provides information about the operation name for the intermediary behaviour, the parameters it takes, including the associated data type information, and the order in which the parameters appear in the signature.

4.8 Summary

This chapter presented our framework. It provides an overview of all the services that make up of VCUDSDC and describes the minimum requirements to allow objects to move around different Environments (both virtual and physical). It describes how our framework is capable of allowing the sharing of digital contents and generates behaviours dynamically (on the fly) using high level semantic, Rules and scripts. In the next chapter, we shall present the case study that successfully demonstrates the VCUDSDC framework discussed in this chapter.

Chapter 5

5 Case Study: Smart Home Environment

5.1 Introduction

A case study is presented in this chapter to demonstrate the functionality provided by our framework that has been implemented as a prototype. The case study explains how a Smart Home Environment can be visualised, where objects (real and virtual) can join the home network and how their visual effects and functionality can be created on the fly in a virtual environment. A bridge between a virtual environment and the physical world is created and used to control physical devices from the virtual world. The case study demonstrates how our framework can be used to generate object functionality on the fly and visualize its appearance in the virtual environment. These objects (behaviour and visualisation) can be discovered in the Home Network and utilised in new and novel ways.

5.2 Case Study

In this section, a Smart Home Environment is presented which visualises, discovered objects and generates object functionality on the fly and allows digital content to be shared with other peers and to discover ubiquitous devices and services hidden in Home Networks. The framework will first discover the best possible behaviour, and its visual data. If it fails to discover or generate behaviours the user is asked to reject that object from the system or keep it until a new behaviour is discovered within the network. Whenever it finds the behaviour then it will notify the user of the newly discovered functionality generated for that object.

The selected case study helps us to test the design decisions presented within the thesis and demonstrates how a Smart Home Environment can be created that will address the key requirement stated in Chapter 1.

Imagine you leave your home to go on holiday and there is no one in. Suddenly you remember that you have forgotten to switch off the lights and you do not have the option to return to your home. It would be nice if we could log in to a virtual representation of our home, using our mobile phone for example to switch the devices off. The challenge is to create functionality in the virtual word for each device. A bespoke application could be created, however this would be too costly and difficult to integrate new devices automatically. A more flexible open framework is required that is connected to the same physical object and its functionality. Devices need to automatically appear in the virtual world without much effort where device functionality is mapped seamlessly between the two. This allows objects to be controlled either virtually or physically.

We have many network enabled devices in our home but currently it is difficult to utilise these devices in full. Imagine a future Home Environment; you buy a Lamp and plug it into the network and your virtual environment detects the device and automatically creates the required functionality and its visual information without any human intervention. When you click the switch off button in the virtual environment the physical Lamp is switched off and vice versa as illustrated in Figure 5.1. This will give you the ability to access physical devices from anywhere in the world.

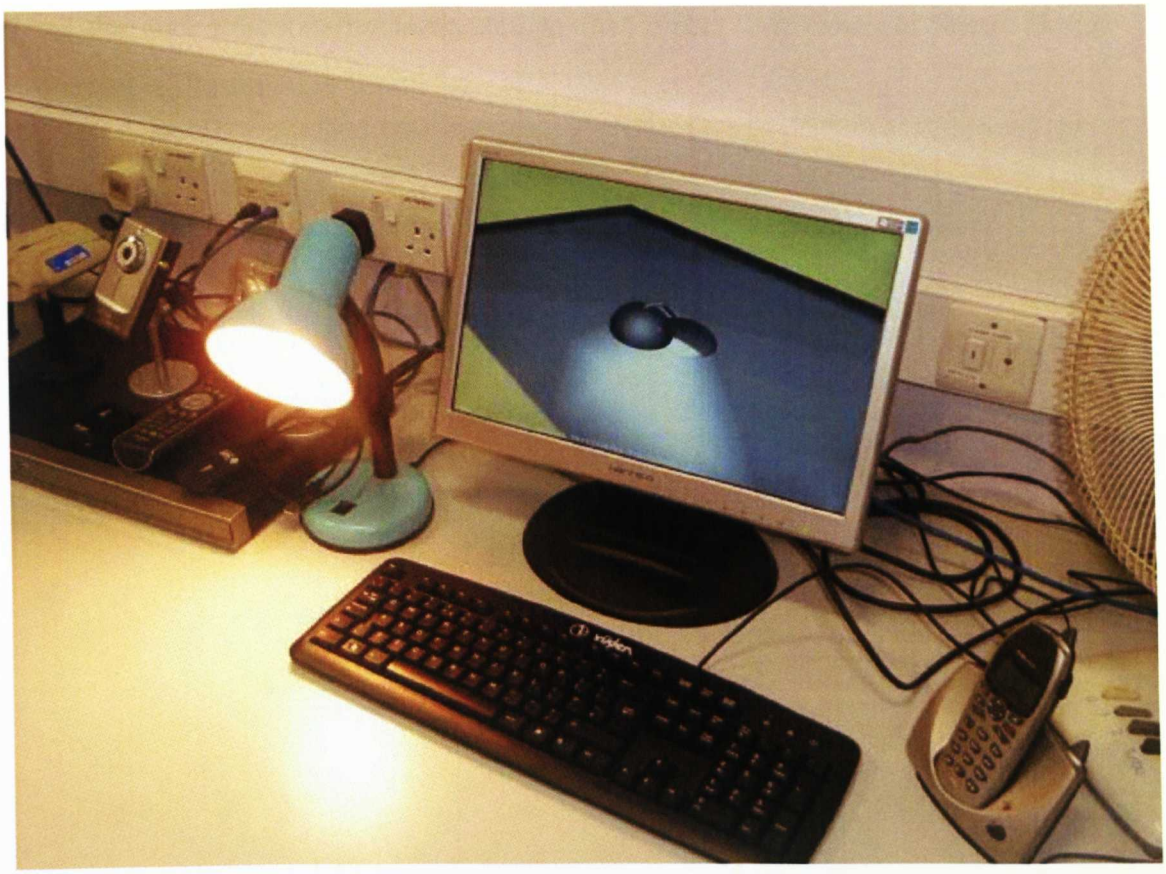


Figure 5.1 Lamp

Taking this vision one-step further, the virtual environment is highly flexible and allows objects (real and virtual) to be added at run time and behaviours to be generated on the fly. If the environment does not have the exact functionality, it will attempt to find it in the network and allow the object to be integrated into the environment without human interaction. Unlike traditional systems, we do not rely on human interaction to configure the device. Imagine if you buy a new device and a user does not have basic programming skills to create device functionality in the virtual world. The virtual world will listen for new objects, when you connect the new device to the network. The virtual environment will find the device automatically and start communication with the device to extract as much information as it can to generate its functionality on the fly along with its visual effects.

Mechanisms allow the virtual world to sense new improved behaviours and update device functionally at run time without human interaction. Returning to our Lamp example, if the flashing lamp functionality moved from another virtual environment then the system will query its entire objects to find appropriate behaviours and notify

users of the new functionality integrated to that object. An example Smart Home Environment is illustrated in figure 5.2.

Figure 5.2 Smart Home Environment

The virtual environment has the provision to provide any number of behavioural services. Once the object has connected to the virtual environment, it keeps track of each object and its behaviour and visual effects through continuous updates. For example, a basic mobile phone provides the functions to make a call, stop a call, answer a call but later the user may buy a new mobile which has more advanced functionality i.e. audio, video and GPS. The virtual world will borrow this functionality from its TV or DVD object and create this functionality for new mobile phone as illustrated in Figure 5.3.

Figure 5.3 TV

Imagine you are on holiday and you left your mobile phone at home. Somebody wants to call you, but you are not able to answer. However, with the help of a computer in your area and Skype software this problem can be solved as illustrated in Figure 5.4. Your friend wants to call you. How can you attend the call? First, he needs a network-compatible computer with the virtual lab installed. The second precondition is a network-compatible mobile phone, which should be no problem today. The functionality generated by the virtual environment for the mobile phone allows it be answered and stopped via the Internet. Your mobile phone wraps these packets into RTP packets, redirects and sends them via the Internet to the virtual world on the computer in France. The mobile phone on the table in the virtual world recognises a call coming in and you can answer the call by clicking the virtual phone button. The RTP packets from the virtual world return mobile packets back to the physical device. Your mobile phone acts like a kind of server as illustrated in Figure 5.4.

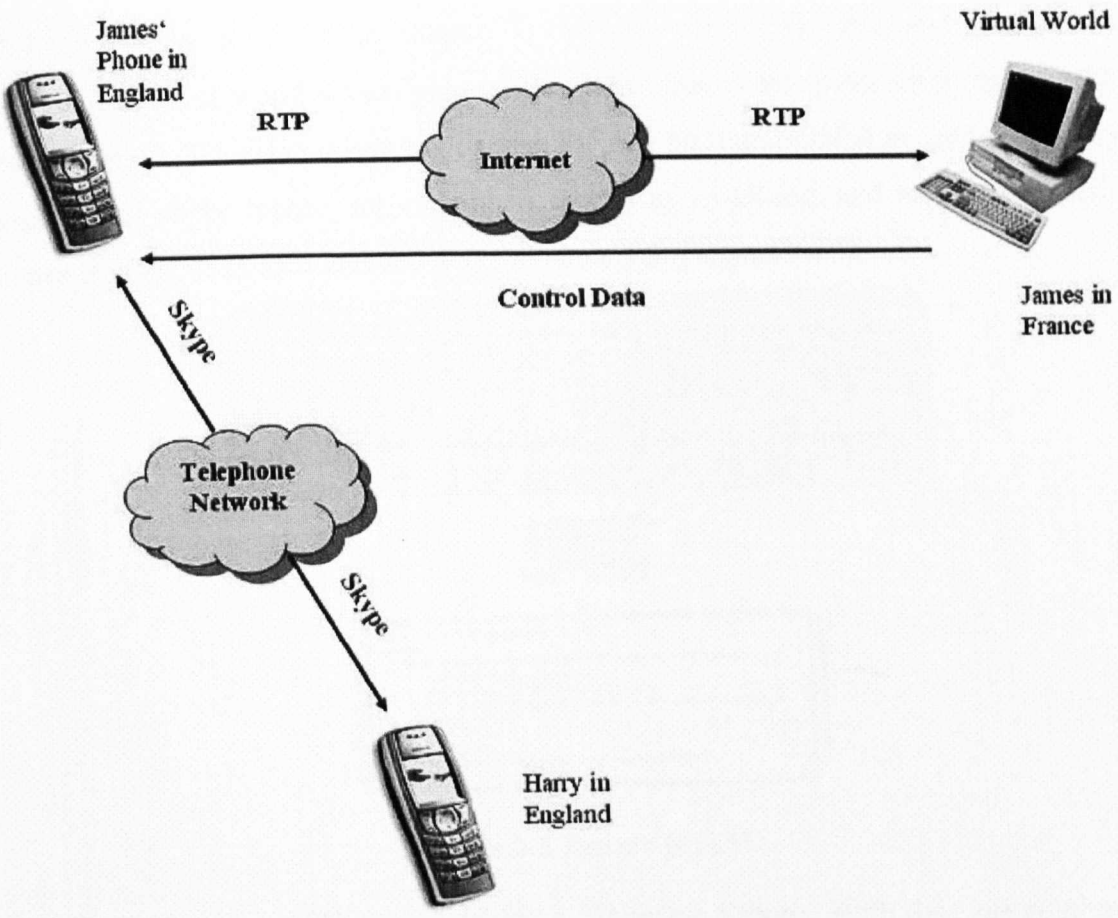


Figure 5.4 Mobile Phone

In this way your friend does not feel that he is connected to a virtual world but instead it feels like he is connected to your mobile phone which is not possible in the real world unless you physically have your mobile phone with you.

The next step is to advance the project in a way that we can present it to a greater audience. That means the real purpose of the project must be clear – not simply just a prototype. Different functions have to be added in order to present more possibilities for the user to interact with the virtual environment and to let the virtual lab look more realistic.

One of the most interesting things is to implement the Universal Plug and Play protocol (UPnP) . With UPnP it is possible to play a video in the virtual lab and in the real world at the same time. Therefore, a media server can be used, for example the D-LINK DSM 520 [D-LINK 2010]. With the help of this media server, we can load a video file from the physical disk, which can then be transferred via an IP based network. The media server can send these data to a TV which is capable of playing the video. At the same time, the video is displayed on the TV screen in the virtual

lab.As we have discussed in chapter 1, we can control devices and manipulate contents in virtual world which give us flexibility that is not possible in real world. For example a 3D video displayed on the TV can be manipulated to grab out every single frame of the movie, for example a man who's walking, and export the single frames to XML files as it is illustrated in Figure 5.5 [Fergus 2008].

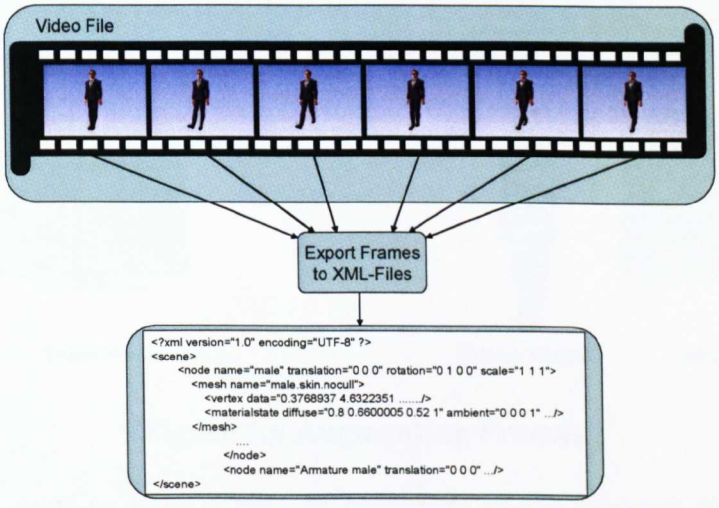


Figure 5.5 Frame to XML

The aim is to have 3D meta data and a 2D frame Engine. Both data are combined in the augmented Engine and subsequently transferred to a network device. These data arrive via a network at the network interface in the virtual environment. The augmented frame engine splits the data into 3D meta data and 2D frame data. The game engine resumes the two different data and renders a 3D scene model and a 2D image on the virtual screen illustrated in figure 5.6.

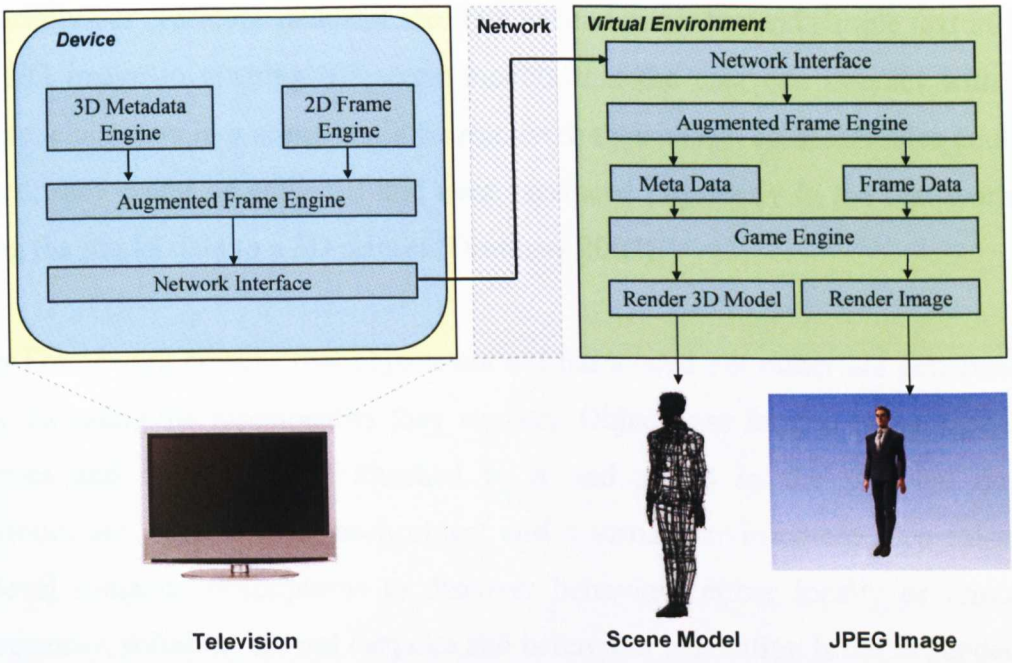


Figure 5.6 Augmenting Frames

In the end we want to have a kind of hologram of the pictures displayed on the virtual illustrated in Figure 5.7. You can move around this hologram and see it from all angles.

Figure 5.7 3D Scenery

Crucially our prototype demonstrates that we can move beyond simple texturing of the JPEG image to creating 3D scene models that the user can interact with. For example, characters in a scene could be removed; their visual characteristics could be changed; they could be collected and even produced physically in the real-world by sending the model data to a 3D printer [Dimitrov 2007].

Our Framework ensures that objects are not hardcoded but rather are generated on the fly including the functionality they support. Objects are formed using high-level semantics and its 3D model attached to it and stored in the physical device. Behaviours are generated for each object and a virtual environment uses rules and high-level semantic descriptions to discover behaviour either locally or remotely. Consequently, solutions are not bespoke and behaviour generation is not dependent on pre-defined variables. Behaviour generation is based on how well the capabilities provided by objects map onto environment requirements.

5.2.1 Characteristics of this study:

Several characteristics are demonstrated within the case study that validates how the UHD prototype works. These characteristics are described as follows:

a) Objects join the virtual environment and their functionality is generated on the fly.

b) Visual effects are generated from their 3D model.

c) Multiple behaviours can be combined to form a single script for each object

d) The virtual environment can simulate real world behaviour and bridge the gap between virtual and physical world objects. For example, a physical lamp could be managed from within the virtual world.

e) New objects and behaviours can be automatically discovered and created that may not have been supported initially. For example, there is no concept of a weapon in a racing game but using high-level semantics we can create a weapon behaviour in a racing game using the common behaviour damage.

These characteristics demonstrate how a smart home can be used to utilise available functionality provided by objects and to generate visual effects and behaviours on the fly.

5.2.2 Using Our Framework for a Smart Home Environment

Several steps need to be taken to configure our Ubiquitous and Virtualisation Home Devices and Service (VCUDSDC) to implement the Smart Home Environment. These are described within this section.

Step 1: Creating The Dynamic virtual environment – In this case study the basic virtual environment has been created and objects are loaded from data sources, in our case, we have used a relational database to store object information like object descriptions, its source code and its 3D model information. The virtual environment implements a Rule Engine to generate object behaviours dynamically for each object using descriptions of an object. The scripting engine provides the functionality to compile code at run time thus allowing objects to be integrated into the environment at run time without recompiling the whole source code.

Step 2: P2P Services – P2P is used in the virtual environment to discover objects and its behaviours remotely within the network. When objects are discovered, the associated behaviours are also imported into the environment and any source code is attached if it is supported by the destination environment.

Step 3: Set of Rules – Set of rules are implemented in our framework to find out the behaviours of particular objects. For example, rules have determined object types i.e. lamp and then other rules are applied to create behaviours.

Step 4: 3D models of objects – We will discuss in more depth how 3D models can be generated in chapter 6 section 6.5.2. In our case, we have used Blender to create 3D models and saved the model as XML this is loaded into Java Monkey Engine to render the object.

Step 5: Scripting Engine: - The script engine evaluates the script and checks its syntax. If it finds any syntax error in the script, it will notify the user. In our framework, several scripting languages are supported to make it more flexible for scriptwriters.

Once these steps have been completed, a combination of rules can be used to create multiple behaviours for each object. For example, a mobile phone will have multiple scripts combined to provide answer, stop, and reject call functionality. Script generation is based on rules and object semantic descriptions the virtual environment propagates semantic queries within the network, which are matched with possible behaviours.

We may not have behaviours at all in the network. In this case, object behaviours are generated and in some cases, multiple behaviours from different objects may exist. For example, a mobile phone, a fan and a lamp all provide "switch On" and "Off" functionality, this can lead to conflict. The semantically closest behaviour out of all behaviours is selected.

5.2.3 Positive aspects of this Case Study

This case study provides a number of advantages over other solutions. Devices can be controlled from virtual avatars and its functionality can be generated automatically using Rules and scripts. This case study illustrates how behaviours can be generated on the fly using Rules provided by VCUDSDC. Many virtual environments are hardcoded and if new objects join, they have to be manually implemented. In the VCUDSDC framework, this process has been automated and device behaviour generated using rules and its semantic description.

5.3 Other Application Domains

VCUDSDC has been designed as a flexible architecture that can be used by a large number of application domains. We have presented a Smart Home Environment solution, however it can be used in other large networked environments whether they are based on infrastructure networks such as LANs and WANs or ad hoc networks whereby structural change is dynamic. Consequently, this section describes some of the application domains in which our framework could be used.

5.3.1 Games

In online games such as *World of Warcraft* [Ducheneaut 2006] and *Second Life* [Herman 2006] players share a virtual environment in order to communicate, do business, and develop digital objects, which not only involve personal computers but

also games consoles and mobile devices. Although no real devices are used (apart from conventional input mouse keyboard) and the user is the only physical entity, these players communicate over large distance from different geographical locations from all over the world. Users can generate and share content, and buy and sell it in virtual environments but there is no middleware which makes it possible to share content across different virtual environments such as Second Life but they allow limited sharing amoung same environments For example you can share object within second life but you can not export to world of warcraft. As such our framework allows the following requirements to be realised.

- VCUDSDC provides an 'intelligent' middleware that allows digital content to be shared across different virtual environments [Shaheed 2007].
- Objects can be used in different environments. VCUDSDC matches common terms between objects in source environments and generates behaviours. This allows us to share content across different environments and generate functionality on the fly.

5.3.2 A 3D Internet Interactive Commerce

Currently human computer interaction is very limited in terms of accessing content on the internet resulting in alternative interactive solutions, such as those used in existing 3D e-commerce sites (www.3DInternet.com). Nonetheless, early adopters of the 3D Internet provide limited interactivity with models (car, phone, or television) beyond simply rotating objects in the 3D space. The challenge is to extend the concept of a browser to include 3D capabilities and to allow for more complex interaction within those containers. This would see solutions even extending the capabilities of current gaming systems where the granularity of detail and interaction far exceeds those of simple avatar control or whole object manipulation. Conventional 3D applications allow the person to be moved and the car to be controlled. New platforms need to be developed that provide all the interactive capabilities as its real-life counterpart.

The VCUDSDC framework allows users to buy or sell products in ways that they have never done before [Amjad Shaheed 2009]. Simply by searching for a product such as car you require. Cars matching your search requirements will be returned as detailed interactive 3D models that are completely operational in the same way a real car is. Each discovered car is dynamically loaded into the 3D desktop. Once a car is loaded and then selected the desktop allows the user to interact with the features of the car, i.e. open the car door, turn on the engine, activate the indicators, turn on the radio and look at all the parts that make up the engine.

5.4 Summary

This chapter demonstrates how our framework can be used to implement a Smart Home Environment, capable to visualising hidden services, operate physical devices from virtual environments, and operate those devices which cannot be operated physically because they are difficult to access. The core functions highlighted within the case study can be adapted and applied to different home networking scenarios. Numerous functionalities can be automatically created on the fly using semantic descriptions. Extending the application domain further this chapter also highlights several other application domains VCUDSDC could be used in.

The case study has demonstrated that our framework is flexible and portable across many different problem domains. It highlights a completely new and novel way to bridge the gap between physical and virtual worlds and surpasses existing middleware solutions. Behaviours are generated on the fly using semantic descriptions and visualising hidden services and dispersing them within the network results in a ubiquitous environment. It will bring many benefits such as in e-commerce solutions where 2D content is transferred to 3D to allow users to play with virtual products and simulate their functionality to provide them with new and novel experiences. Furthermore, it will bridge the gap between physical and virtual environments making interaction with content, services and devices seamless.

Chapter 6

6 System Implementation

6.1 Introduction

In this chapter, we present the implementation of our framework described in Chapters 4. This chapter begins by describing the goals of our framework in relation to networked appliances. The framework is an example of a service-oriented architecture (SOA) and therefore addresses the same objectives. The individual services our framework provides are described in detail, which also includes a description of the prototype we have developed to evaluate our framework design.

6.2 Framework Architecture

VCUDSDC (Ubiquitous and Virtualisation Home Device and Services framework) is a service-oriented architecture. It provides mechanisms to visualise devices and behaviours in a virtual environment. This provides more control for devices that may not be possible physically such as accessing temperature data from sunSpot. Chapter 2 and 3 introduced the common concepts used within home networking, networked appliances, peer-to-peer computing, virtual environments, scripting languages, Rule engines and the semantic web. The implementation uses these concepts throughout this chapter to realise the novel contributions detailed in chapter 1.

6.3 Framework Services

The following subsections discuss the implementation details for each of the services used to implement the VCUDSDC framework. A discussion is presented on the technologies used to achieve this, which includes the benefits they provide, the difficulties we encountered and how they have been extended to incorporate our novel contributions. The framework illustrated in Figure 4.1 and 6.1 shows the services used within the VCUDSDC framework and the relationships that exist between them.

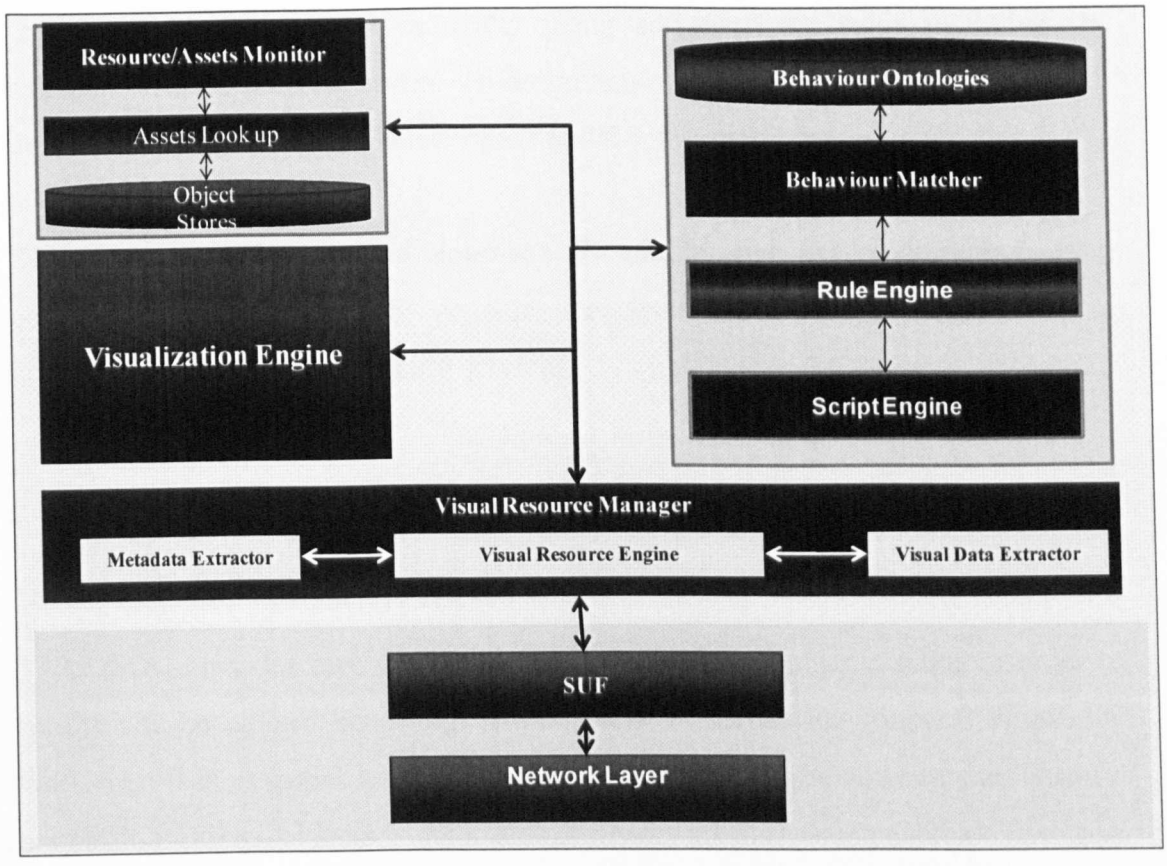


Figure 6.1 VCUDSDC Framework

The remaining subsections discuss the key techniques used to implement VCUDSDC that includes the Visual Resource Manager, Visualisation Engine, Rule-Based System, Scripting Engine, Behaviour Matching, Behaviour Discovery, SUF (Service Utilisation Framework) peer-to-peer network.

6.4 Visual Resource Manager

VCUDSDC provides a technique to share digital contents across heterogeneous environments and allows virtual and physical components to control these objects from within virtual worlds. Our framework set on top of SUF (Service Utilisation Framework) [Majid 2008] which provide P2P services such as to to discover new objects within the network (SUF uses JXTA protocols). These protocols allow any object to be shared regardless of the environment, programming language, platform, or the transport protocols objects implement.

VCUDSDC secondary services have been developed within the service layer of the VRM (Visual Resource Manager). This allows objects to be discovered, behaviours to

be matched and generated dynamically using scripting and rules including the visualisation of objects in virtual environments. VCUDSDC secondary services extend the SUF framework to include these additional capabilities.

SUF uses a service-oriented approach to enable inter-object communication, discovery and invocation. SUF provides pre-determined pipe advertisements to discovered objects and its secondary services.

6.5 Secondary and Application Specific Services

6.5.1 Asset Lookup and Resource Monitor

VCUDSDC provides mechanisms to keep track of each objects status in order to allow them to be utilised better e.g. if a wireless sensor device connects within the network then it is essential to keep track of its battery life. Because of the dynamic nature of VCUDSDC, objects with various attributes are stored in a database (we have used Microsoft Access with JDBC). These attributes are retrieved whenever they are needed. When objects have been found and their behaviour and visual effects have been created, the Asset Lookup service registers the object's properties. Another function of this service is to keep track of the object status in order to determine whether the asset is sharable or not. If it is sharable then it determines whether it can be used. For example, if the asset is a sunSpot then it checks whether the sunSpot has enough battery power, if it does not then the Asset Lookup informs the Visual Resource Manager that the object is not available. The Asset Lookup module stores the location of the script path and its visual effects path in the database to make sure the objects behaviour and its visual effects are loaded next time.

6.5.2 Visualisation Engine

VCUDSDC uses the JME games engine to render objects in virtual environments as illustrated in Figure 6.2. Blender has been used to design the 3D models. As can be seen in Figure 6.2 the grain of the wood on the floor, soft shadows (near the table), smooth lights and exact physical movement and also the chrome-plated doorknob on the right side reflects the light.

Figure 6.2 Room view in Blender

6.5.2.1 Export to XML Files

After the complete model is created it can be rendered. Of course it is possible to make modifications afterwards. If the result is extensively satisfying the model must be converted into a format usable for high level programming languages. JME based on Java uses the Extensible Markup Language (XML) format. Java and XML are platform independent. Both Java and XML complement one another in a distinguished way and together they are used more and more in developing large systems. Simple configuration files are read in, certain system conditions are stored or complex data between different systems are exchanged. The schema for javaMonkey engine is shown in Figure 6.3.

```
<scene>
<node>
<mesh>
  <submeshes>
    <submesh>
      <faces>
        <face>
                    <face/>
      </faces>
      <geometry>
        <vertexbuffer>
          <vertex>
                        </vertex>
        </vertexbuffer>
      </geometry>
    </submesh>
  </submeshes>
</mesh>
</node>
</scence>
```

Figure 6.3 XML schema of the mesh file

The name of the node and the position is shown in line 3 of Figure 6.4. The vertex data in line 6 indicate the position of every single vertex of the window in the three-dimensional space. Line 10 illustrates the material state of the window and line 12 indicates which picture is used as a texture which means that, *landcape.jpg*' will be seen in the implementation.

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <scene>
3          <node name="Window" translation="0.2157859 9.4397106 -66.4921188" rotation="0 0.7071068 0.7071068 0" scale="2.5246232 1 3.1637747"
4          >
5              <mesh name="Window1.nocull">
6                  <vertex data="-10.1648264 0 5.3645916  10.1648216 0 5.3645935  10.1648264 0 -5.3645935  -10.1648264 0 -5.3645935  "/>
7                  <normal data="0 1 0  0 1 0  0 1 0  0 1 0  "/>
8                  <texturecoords data="0 1  0 0  1 0  1 1  "/>
9                  <index data="0 1 2 2 3 0"/>
10                 <materialstate diffuse="0.8 0.8 0.8 1" ambient="0 0 0 1" emissive="0 0 0 1" specular="0.25 0.25 0.25 1" shiny="12.5244618"
                   alpha="1"/>
11                 <texturestate  >
12                     <texture file="landscape.jpg" texnum="0" wrap="3"/>
13                 </texturestate>
14             </mesh>
15         </node>
16 </scene>
```

Figure 6.4 Scence 3D model data for window

6.5.2.2 3D Gaming Engine

The task of a 3D engine is to manage all 3D objects and scenes. In addition the engine interprets what is displayed on the screen and how it is drawn. We have put a two-dimensional picture on the screen from the information provided by 3D objects, a virtual camera, textures and lights is the purpose of the rendering pipeline. The 3D engine wraps objects with a given 3D interface using OpenGL and abstracts the so-called rendering pipeline.

A 3D pipeline is filled with 3D data which can be changed while running through the pipeline. The result at the end of the pipeline is a complete picture on the screen. In order to render the laboratory in real time it is useful to use a 3D engine. Real time rendering means that every picture is rendered in split seconds in order to move freely in the three-dimensional world. It also means that instructions relocated in the virtual world are executed nearly simultaneously in the real world and vice versa. As a 3D engine we use JME, which is completely written in Java.

The virtual world with all its 3D objects is called a scene. To relate all these objects a so called scene graph is used. In JME this scene graph equals a tree with a root and miscellaneous elements. Every element can have multiple child elements but only one father element. With 'rootNode' JME already provides the root for the scene graph.

In JME three classes exist, which can be elements of a scene graph. These classes are *Spatial*, *Geometry* and *Node*. The Spatial class is the upper class, which cannot be instantiated. It is an abstract class. Location and rotation are stored in here. That means that every object of the scene graph has its own position, relative to the parent element. Setting *RenderStates* is also possible, in order to describe lights and textures. *Geometry* class and its subclasses consist of triangles, 3D objects consist of triangles. Every object visible on the screen is a geometry object. But these objects are just ending elements of the scene graph tree. All nodes are managed by the class Node. Any child nodes and child geometry nodes can be added to this class. The scene graph is the backbone of JME.

6.5.2.3 XML to Scene Graph

JME doesn't support loading any file formats directly. Rather it has JME binary, its own format. It is important to understand file loading with JME. Different classes included in JME convert the according format to JME's binary. First of all the binary converter and binary reader is loaded as it is illustrated in Figure 6.5 from lines 4 to 7. For converting JME works with streams only. As illustrated in line 10 and 12 the OutputStream and the InputStream are used to send and read the appropriated contents. The XML file is converted with the ByteArrayOutputStream and read with the ByteArrayInputStream when the virtual environment starts the Try/Catch block is used to handle potential errors.

```
1  private Node LoadXML(String filePath){
2      try{
3          //  Initialise binary converter
4          XMLtoBinary converter = new XMLtoBinary();
5
6          //Initialise binary reader
7          JmeBinaryReader jbr = new JmeBinaryReader();
8
9          // convert the input file to a jme-binary "LabCube"
10         ByteArrayOutputStream LabNoExtra = new ByteArrayOutputStream();
11         URL LabModel = ModelLoader.class.getClassLoader().getResource(filePath);
12         converter.sendXMLtoBinary(new BufferedInputStream(LabModel.openStream()), LabNoExtra);
13
14         //get the "LabCube"
15         tempNode = new Node ("Temporary Node");
16         tempNode = jbr.loadBinaryFormat(new ByteArrayInputStream(LabNoExtra.toByteArray()));
17     }
18     catch(IOException e){
19         System.out.println("Couldn't load the input file:" + e);
20         e.printStackTrace();}
21     catch (java.lang.NullPointerException npe ) {
22         npe.printStackTrace();
23     }
24     return tempNode;
25 }
```

Figure 6.5 XML to JME

Figure 6.6 shows the implementation of a Media Player to load data from an XML file into JME. In line 2 the function *LoadXML* is used with the appropriated XML file "*dvd.xml*'. Line 5 ("*attachChild*') illustrates the attachment of the media player to the virtual environment. The virtual environment becomes the parent of the media player. But there are some more interesting methods implemented in this piece of code. In line 3 of Figure 6.6 we can see the methods ,*setModelBound (new BoundingBox())*'. In the area of 3D graphics different coordinate systems are used and in particular between the so-called world space and model space. Every object in a 3D world has

its own local coordinate system with its own axes and point of origin. The point of origin for example is in the centre of the object. All axes indicate which directions are '*top*', '*bottom*', '*right*','*left*', '*front*' and '*back*' for the object. To put the object, in this case the media player, into a coordinate system like this a '*BoundingBox*' (line 3) is required. The media player now has its own coordinate system, the so-called ‚*ModelBound*', *UpdateModelBound*' in line 4 recalculates the bounding box for the media player. Of course the whole scene or world also has a coordinate system, the absolute coordinate system. Every place in the 3D world can be described as a set of the x-, y- and z-coordinates. So the media player has its place in the virtual lab. In line 6 in Figure 6.6 you can see, *updateWorldBound*'. It merges the bounds of all the children maintained by the media player. This allows faster culling operations.

```
1  private void setupMediaPlayer(){
2      media = LoadXML("dvd.xml");
3      media.setModelBound(new BoundingBox());
4      media.updateModelBound();
5      lab.attachChild(media);
6      media.updateWorldBound();
7  }
```

Figure 6.6 Media Player Object

Figure 6.7 describes the XML file of the media player. As you can see above the media player is the child of the virtual environment. The virtual environment itself has the rootNode as its father. Line 2 of figure 6.7 shows the *play* button as a child of the media player scene. The *stop* button in line 5, the *forward* button in line 8 and so on are also children of the media player. That is exactly what is described in the scene graph.

```
<scene>
    <node name="Play" translation="..." rotation="..." scale="...">
        ...
    </node>
    <node name="Stop" translation="..." rotation="..." scale="...">
        ...
    </node>
    <node name="Forward" translation="..." rotation="..." scale="...">
        ...
    </node>
    <node name="Rewind" translation="..." rotation="..." scale="...">
        ...
    </node>
    <node name="Pause" translation="..." rotation="..." scale="...">
        ...
    </node>
</scene>
```

Figure 6.7 Media Player in XML

Figure 6.8 illustrates the hierarchy relationships of Media Player and its different functions

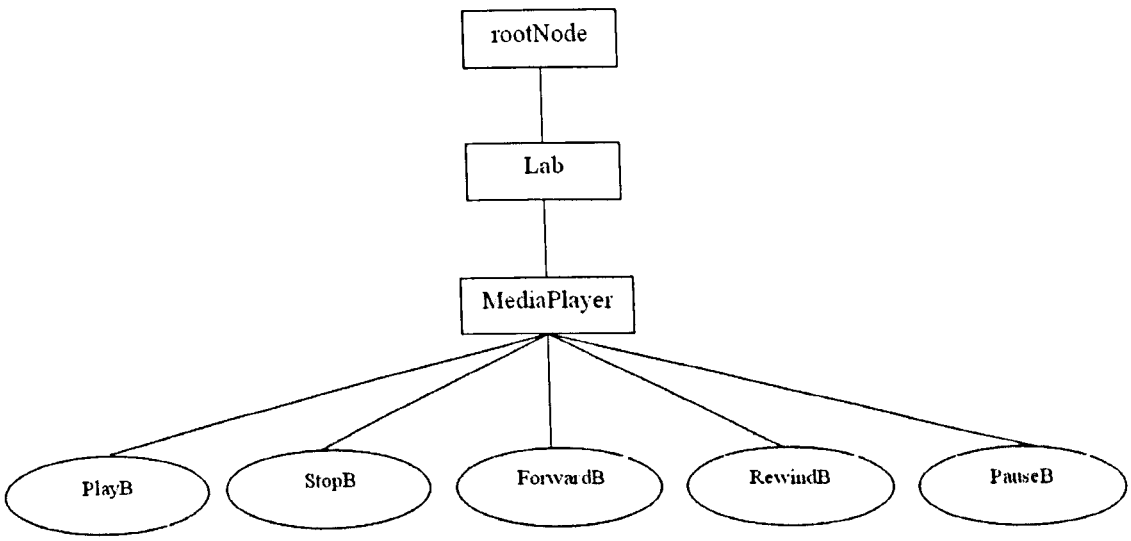


Figure 6.8 Hierarchy of the Media Player

6.5.3 Scripting Engine

The scripting Engine is responsible for providing dynamic behaviours in virtual environments where objects can be added at run time and behaviours generated on the fly. As we have discussed the requirements for the script parser and generator in Chapter 4 section 4, provide multi language support, [mikeg 2008]. For example, JavaScript, jRuby and many more. The script parser determines which script language we are using and checks the appropriate syntax structure as shown in Figure 6.9. If errors are found an error exception is thrown and the script is not generated.

The code shown in Figure 6.9 first loads the right scripting language in order to run that script using that language. For example, Java Script in Figure 6.8 is used. We have given flexibility to choice any scripting language instead of restricting it to one single script language. Environments that do discover and support a particular language can automatically load required scripting languages.

```
public Script(StringscriptFile) {

    Compilable eng = null;
    scriptFileName = scriptFile;
    initEngine();
    try {
      /**
       * If script iscompiliablewe compile it.
       */
      if (engine instanceofCompilable) {
        // Compile script
        eng = (Compilable) engine;
        BufferedReader fr = new BufferedReader(new FileReader(scriptFileName));
        String line;
        while((line=fr.readLine()) != null) {
          scriptContents.append(line+"\n");
        }
        scr = eng.compile(new FileReader(scriptFileName));
        engine = scr.getEngine();

      }
      //set engine scope namespace
      engine.setBindings(n, ScriptContext.ENGINE_SCOPE);

    } catch (FileNotFoundExceptionfnf) {
            Logger.getLogger("cigame").log(
          Level.SEVERE,
          "Script file not found: " +scriptFileName,
          fnf);
    } catch (ScriptExceptionse) {
            Logger.getLogger("cigame").log(
          Level.SEVERE,
          "Script exception: " +scriptFileName,
          se);
    }
    catch (IOExceptione) {
            Logger.getLogger("cigame").log(
          Level.SEVERE,
          "Script read error: " +scriptFileName,
          e);
    }
    if(engine instanceof Invocable) {
      inv = (Invocable) engine;
    }

}
```

Figure 6.9 Scripting Engine

Once the script has been created and assigned to each object in the virtual environment and then evaluated by the scripting Engine if script contains any error shown in Figure 6.10.

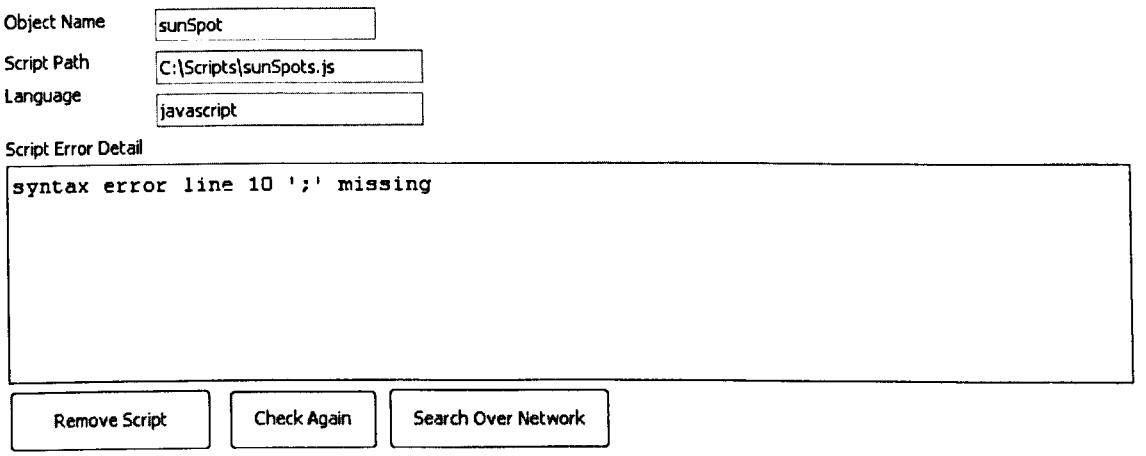


Figure 6.10 Scripting Error Box

Figure 6.11 illustrates the code for evaluation process taken by the scripting engine. This process will allow us to check for syntax errors in the script before assigning the script to the object.

```
/**
  * Evaluate the script.
  *
  * NOTE: This will revaluate the script even if it has already been
  * evaluated.
  *
  * @return
  */
  public Object eval() throws ScriptException {

    try {
      evaluate();
      evaluated = true;
    } catch (ScriptException e) {
      // TODO Auto-generated catch block
            Logger.getLogger("cigame").log(Level.SEVERE,
          "Error evaluating the script " + this.scriptFileName, e);
      throw e;
    }
    return evalReturnObject;

  }
```

Figure 6.11 Script Evaluation

Constructing the scripted behaviour is an iterative process, which could result in additional behaviours being added to the object for example, the implementation of SunSPOT sensors provides three main behaviours such as *GetLight, GetTemperature,* and *GetAccelerometer* as shown in Figure 6.12.

Figure 6.12 LJMU Network appliances Lab

SunSPOT sensing devices were used in the prototype. Two types exist. The first type is the base station and the second is the free range. The base station is connected to a laptop (this allows the laptop to form part of the ubiquitous computing network). Services exist on free range spots that allow access to the light, temperature and accelerometer functions, via the base station. Each free range sensor in the ubiquitous computing environment can calculate light, temperature, and acceleration values. It is this data that is streamed to the cube in the virtual environment and depending on the values changes are made to the cube, i.e. change colour or move position. Using this prototype and our previous work [Shaheed 2007] sensors and their associated functionality can be projected into the virtual container to form a bridge between the physical and virtual object. Irrespective of the size of the sensor used our proposed platform allows us to still see what devices look like and what functions they provide.

6.5.4 Rule Engine

Here the rules try to extract the behaviours the target environment supports. The Behaviour Ontology acts as an interoperability mechanism between terminologies which we have implemented and serialised using the Web Ontology Language (OWL) [Berners-Lee 2001]. Rule engine has been discussed briefly in chapter 2, the structure of Rule is illustrated in Figure 6.13.

```
Rule "Rule Group"
When Condition
.....
Then
Action.....
End
```

Figure 6.13 Rule Syntax

The rules were developed using Drools [Drools 2011], as illustrated in Figure 6.4. This example shows a simple SunSPOT script to create three types of functionality: first to obtain environmental light information, second to obtain Temperature information and third to get 3D accelerometer information. The description information about SunSPOT is stored in the database, which allows us to fire rules and create the script for the SunSPOT sensor. Once the right script has been created for an object, it is saved to a physical location as shown in Figure 6.11 and the object ID is set to the file name and finally the UpdateSourceCode property is set to false in order to avoid rules firing continuously.

```
rule "Object Name SunSpot"
        salience 0
when
        object:gameObject(scriptEnable==true,
                                                                name=="SunSPOT Sensor",
                                                                Function="Temperature,Light,Accelerometer"
                                                                updateSourceCode==true)
        #conditions
    then
        String Code=getTemplateCode(url);
         Code=getSunSPOTVariables();
        Code=Code + getInitializeSunSPOTVariable();
        Code=Code + getLightLogic();
        Code=Code + getTemperatureLogic();
        Code=Code + getAcceloremterLogic();
        String fileName=url.replaceAll("BehaviourTemplate.js", object.getId() + ".js");
        rewriteBehaviour(Code,fileName);
        object.setScriptFileName(object.getId() + ".js");
        #actions
End
function String getLightLogic()
{
String ret="dg2.writeUTF("Send Me Light");    conn2.send(dg2);    conn2.receive(dg2);    System.out.println(dg2.readUTF());    dg2.reset();";
return ret;
}
function String getTemperatureLogic()
{
String ret="dg2.writeUTF("Send Me Temp");    conn2.send(dg2);    conn2.receive(dg2);    System.out.println(dg2.readUTF());    dg2.reset();";
return ret;
}
function String getAcceloremterLogic()
{
String ret="dg2.writeUTF("Send Me Acceloremeter");    conn2.send(dg2);    conn2.receive(dg2);    System.out.println(dg2.readUTF());
dg2.reset();";
return ret;
}
```

Figure 6.14 Rule to generate SunSPOT behaviour

We have demonstrated in Figure 6.14 the generation of behaviours for a SunSPOT sensor. For example, when we move the physical SunSPOT sensor up or down then its associated virtual object also starts moving in an up or down direction as shown in Figure 6.14. If we press the Light Button then it changes the light of the virtual SunSPOT and the temperature button is pressed turns the cube to red in the virtual environment shown in Figure 6.14.

6.5.5 The SUF Peer-to-Peer framework

The main purpose for using SUF (SUF is based on JXTA protocol) is to utilise the Peer-to-Peer services if offers. Objects can be searched and used or the behaviours of objects can be discovered and generated dynamically. SUF gives us functionality to advertise objects and their behaviours and to register objects and its resources. This allows us to search objects using high-level semantics embedded into advertisement messages. Each object in the virtual environment can be discovered using its corresponding advertisement. Each object in the virtual environment is assigned a unique Peer ID to be shared and discovered. Each virtual environment in our implementation advertises its contents, such as objects and their behaviours, in order to allow others to utilise its services. Different Services have been implemented using

Java such as the Object Lookup, Behaviour Looking and Basic Peer's security Services.

Each virtual environment publishes its function as SUF Peer services and allows objects within P2P networks to be discovered and shared. SUF gives us advanced functionality to discover physical devices and zero configuration to allow us to visualize devices in virtual environments.

how the Virtual Lab can be started and stopped, listens for new objects (both physical and virtual) in the network and where it sends messages. To search object behaviours in remote locations the virtual environment propagates messages over communication pipes and once it matches object behaviours, it adds it to a service advertisement. Once all the required behaviours have been found then the Virtual Lab renders the object and attaches behaviours to that object.

6.5.6 Behaviour Matching Services

In our implementation, we have developed Ontologies in Protégé and converted them into Java classes for use in knowledge sharing. Instances of the concepts and relationships between concepts are stored in a knowledge repository through the Protégé JDBC database back-end. This allows fast and efficient updating and querying of the ontology outside the Protégé environment by different components of the application.

Ontologies developed in Protégé are used together with the Drools inference engine (http://labs.jboss.com/drools/) for rule-based reasoning and knowledge acquisition. Using the rules representation, it is possible to match the behaviour of new objects/characters in a new environment with local expected behaviour. This matches object/characters with the closest behaviour or extends the ontology to include new behaviours. The execution environment provides updates on changes in the environment as well as the status of object/characters and human players. These changes are updated into the knowledge-base and the inference engine and the behaviours of individuals and modified accordingly based on the cognitive model.

6.6 The Framework prototype

In order to evaluate our framework design presented in Chapter 4, a prototype has been developed. This is in accordance with the case study presented in Chapter 5, which is a Smart Home Environment. The prototype uses three computers (Computer A, Computer B, Computer C) connected through a LAN network, a Fan connected to Computer A, a Lamp connected to Computer B, and a SunSpot sensor and Skype phone connected to Computer C.

In each computer we have installed and configured our Virtual Lab. In the Virtual Lab we have implemented SUF services which propagates service requests between computers using SUF. When the Virtual Lab is started and the services have been published, they automatically try to discover objects within the environments they have a relationship with. For example, when Virtual Lab A on Computer A is started and the Lamp connected to Computer A tries to discover the behaviours that are capable of switching the Lamp OFF and ON. Figure 6.16 illustrates the user interface to discover and transfer the object connected to the network and the services it provides. Once the object has been selected for transfer, the system sends Meta data information as an XML file through to the destination system (Computer A).

Figure 6.15 Virtual Lab User Interface

Two tests have been developed to evaluate VCUDSDC. The first test demonstrates that VCUDSDC can allow objects to move from one virtual environment to another and generate their behaviours without any human intervention. The second demonstrates how to connect physical devices to virtual environments and generate behaviours dynamically. In the first test, we publish the Lamp object services, which are created using the OWL-S Service Profile. Once the user starts to transfer the virtual lamp from Computer B, Computer B starts transferring the Meta data about the Lamp to Computer A which includes visual and behavioural information. Once Computer A receives the Meta data, the visual resource manager starts to separate the visual information from the behavioural information and transfers the visual information to the Visual Engine to render the object. The behaviour matcher service processes the Meta data and discovers behaviours locally using the behaviour matcher class (in our case the behaviour is not found locally for the Lamp in Computer A). Computer A propagates the behaviour request using the OWL-S Service profile within the network using SUF and adds any response to a table of candidate behaviours, categorised according to the type of object discovered.

In the second test, we have connected a SunSPOT sensor device to Computer C where a virtual lab is already running as illustrated in Figure 6.17. We have generated the SunSPOT's behaviours dynamically after sending a request via the network to Computer D, which already has the required SunSPOT behaviours.

Figure 6.16 Controls SunSPOT from Virtual Lab

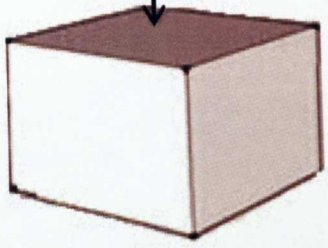
6.6.1 Technical Description

In the architecture described above we have designed a distributed service-oriented platform to link between networked appliances and associated avatars in virtual environments. We have been able to carry out experiments using our design and show how multimedia and gaming content can be shared inside virtual environments. Using JXTA [JXTA 2010] as its peer-to-peer middleware protocol, a virtual environment developed using jME [Engine 2010] queries the network for JXTA services advertised by the peers (Physical mobile phone). We have connected two virtual environments using JXTA.

In the above implementation, a peer makes a request for a service, such as a SunSpot object in the virtual lab where another peer has previously advertised its sharable assets using JXTA advertising services. In Figure 6.18 we show, how we have implemented the scenario in which the user requests a SunSpot sensor resource.

Physical SunSpot Device

```
<?xml version="1.0" encoding="UTF-8" ?>
<scene>
    <node name="SunSpot Sensor" translation="-69.1353256 9.2631273 -0.0025484"
rotation="0.5 0.5 0.5 -0.5" scale="1 1 1.3021666">
      <mesh name="Plane.nocull">
        <vertex data="-16.132164 0 6.7143831 16.1321564 -1e-006 6.714386
16.132164 0 -6.7143841 -16.132164 1e-006 -6.714386 "/>
        <normal data="0 1 0 0 1 0 0 1 0 0 1 0 "/>
        <index data="0 1 2 2 3 0"/>
        <materialstate diffuse="0.8 0.8 0.8 1" ambient="0 0 0 1" emissive="0 0 0 1"
specular="0.25 0.25 0.25 1" shiny="12.5244618" alpha="1"/>
      </mesh>
    </node>
</scene>
```



Virtual SunSpot Object

Figure 6.17 Rendering information for a sunSpot Sensor

We simply pass the meta-data to jME, which in turn is used to render a 3D representation of the SunSpot object in the scene. The SunSpot object also contains the scripting behaviours it supports. For example, Figure 6.19 illustrates, in part a simple script for the *get temperature* behaviour. Javascript was used and were developed using the Rhino API from Mozilla [Mozilla.org 2007], which is used with the Java Scripting Framework [O'Connor 2006] and the open-content repository API provided by Captive Imagination [Captive Imagination 2007].

The Meta data and scripts, including the aforementioned tools where applied in the same way to allow music to be shared between our mobile phone and its associated

avatar in the virtual lab. The goal here is to show how two very different types of content can be shared. One associated with multimedia and the other associated with conventional game playing objects. Perhaps these act as two extremes between which many other possibilities are possible.

Both meta data for objects and the scripted behaviours are passed between different environments using JXTA pipes and messaging services in which all required information is presented to extract and construct the associated object. Whilst, we simply use the meta data to construct the objects, we run all scripting behaviours through a set of rules, as discussed in the above section on the Behaviour Matcher. Whilst objects may support behaviours in their source environment, it is not necessarily the case in the target environment. Here the rules try to extract the behaviours the target environment supports. The *Behaviour Ontology* acts as an interoperability mechanism between terminologies which we have implemented and serialised using the Web Ontology Language (OWL) [Berners-Lee 2001]. The rules were developed using Drools, where 6.14 and Figure 6.19 shows a simple *get temperature* rule and part of the script for a behaviour being generated.

```
rule "Object Name SunSpot"
        salience 0
when
        object:gameObject(scriptEnable==true,
                                                        name=="SunSPOT Sensor",
                                                        Function="Temperature,Light,Accelerometer"
                                                        updateSourceCode==true)
        #conditions
    then
        String Code=getTemplateCode(url);
        Code=getSunSPOTVariables();
        Code=Code + getInitializeSunSPOTVariable();
        Code=Code + getLightLogic();
        Code=Code + getTemperatureLogic();
        Code=Code + getAcceloremterLogic();
        String fileName=url.replaceAll("BehaviourTemplate.js", object.getId() + ".js");
        rewriteBehaviour(Code,fileName);
        object.setScriptFileName(object.getId() + ".js");
        #actions
End
function String getLightLogic()
{
String ret="dg2.writeUTF ("Send Me Light"); conn2.send(dg2); conn2.receive(dg2); System.out.println(dg2.readUTF()); dg2.reset();";
return ret;
}
function String getTemperatureLogic()
{
String ret="dg2.writeUTF ("Send Me Temp"); conn2.send(dg2); conn2.receive(dg2); System.out.println(dg2.readUTF()); dg2.reset();";
return ret;
}
function String getAcceloremterLogic()
{
String ret="dg2.writeUTF ("Send Me Acceloremeter"); conn2.send(dg2); conn2.receive(dg2); System.out.println(dg2.readUTF());
dg2.reset();";
return ret;
}
```

Figure 6.18 Rules used to create scripted behaviour

In the mobile phone scenario we demonstrated how two users were able to share multimedia content between a physical mobile device and corresponding avatars in the virtual world. We stream multimedia content from the physical mobile device to the virtual mobile using the Java Media Framework (JMF) [JMF 2008]and the Real-time Transmission Protocol (RTP) [RTP 2008].

RTP packets were wrapped in JXTA [JXTA 2010] messaging objects to abstract the IP dependent format used for HTTP calls in RTP. This provides a unified addressing scheme ensuring that all components are addressed in a uniform way. Frames were sent from the physical mobile to the virtual environment using JXTA Pipes. Upon receiving the JXTA packets, the RTP packets are extracted and processed by a custom data source adapter developed for the purpose, which streams RTP data much as it is done traditionally. After network connectivity, the avatar requests the list of songs in which he/she can then chose from the list, after that the mobile starts streaming the content using RTP protocol. When it receives the first stream JMF processes the stream and checks for supported codecs. If it is supported then it will continue receiving streams from mobile devices in our case while playing the stream using JMF and a plugin called Fobs4JMF [Ott 2006] which supports most formats such as mp4 or 3gp. These tools in conjunction with Skype allowed us to enable bi-directional communications between the physical mobile phone and its virtual world counterpart. Figure 6.19 shows the code, which has been generated for the mobile phone to make a call using Skype.

```
InputAction buttonAction = new InputAction() {
    public void performAction( InputActionEvent evt ) {
        if (makeCall && evt.getTriggerPressed()){
            keyEvent = evt.getTriggerCharacter();
            keyNumber = evt.getTriggerIndex();
            if((keyNumber == 28) || (keyNumber == 156)){
                makeCall = false;
                try {
                    mc = Skype.call(Number);
                    Number = "";
                    if(!endCall) endCall = true;
                } catch (SkypeException e) {
                    e.printStackTrace();
                }
            }
            else if (keyNumber == 14){
                Number = "";
                list.clear();
                listCount = 0;
                SkypeMsg.print("Deleted");
                return;
            }
            list.add(keyEvent);
            Number+=list.get(listCount).toString();
            System.out.println("Called Contact: " + Number);
            SkypeMsg.print("Calling: "+Number);
            listCount++;
        }
    }
};
```

Figure 6.19 Mobile phone Script to call using Skype

The next appliance to be controlled is the media player. There are several functions the media player can assume. The first idea is to have all of the functions that are offered by a real DVD (Digital Versatile Disc) player simply combined in the virtual world. This means not only starting and stopping a video but also going forwards and backwards through file folders or the video itself. Using rules we have created functionality for the media player on the fly, such as play, stop and exit as illustrated in Figure 6.20.

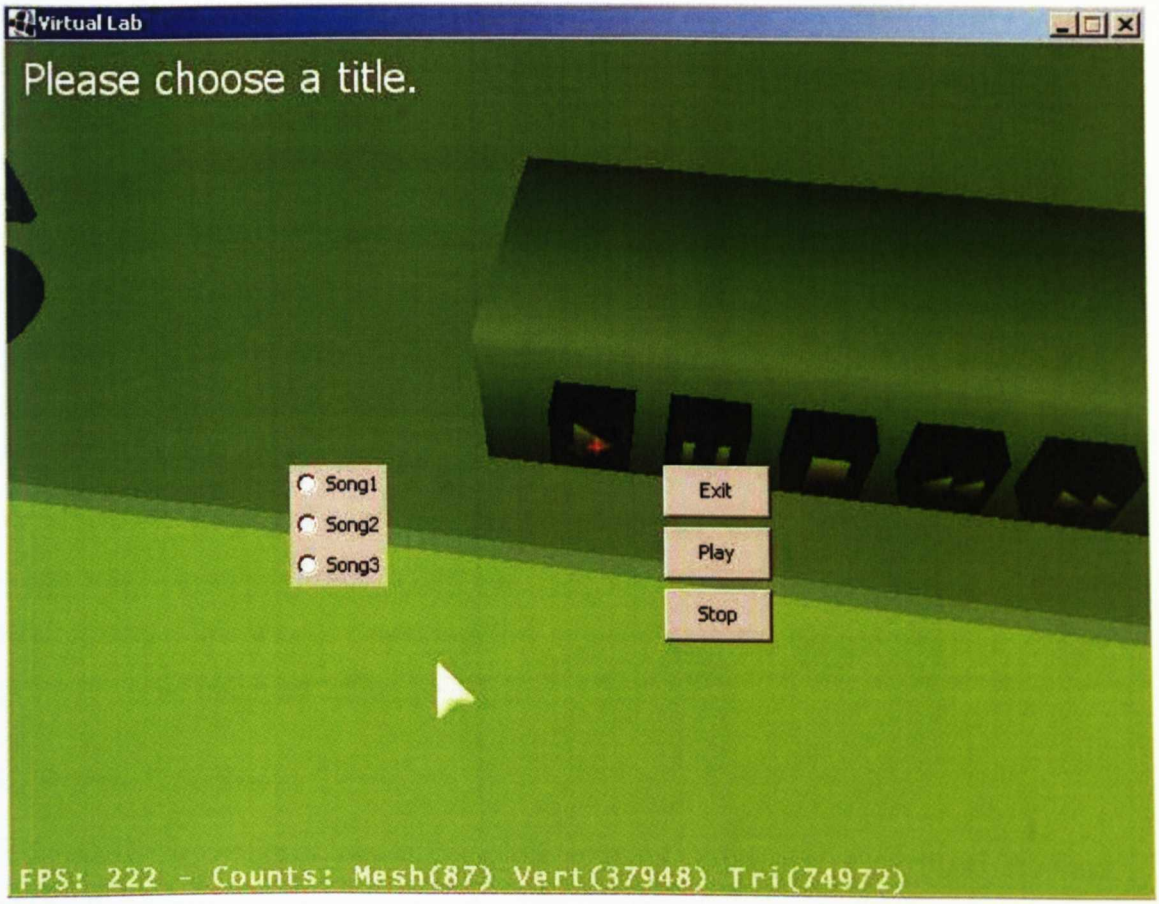


Figure 6.20 Media Player

Ultimately, the success of a framework such as this relies on the development of exciting content that can be used to build up gaming environments. Nonetheless, we believe that a flexible and distributed system such as this provides many opportunities for the advancement of gaming, virtual environments and networked devices in the physical world into new areas and in new ways.

6.6.2 Prototype Configuration

We have setup a prototype within the School of Computing and Mathematical Sciences at Liverpool John Moores University to evaluate the implementation of VCUDSDC. The following off the shelf components have been used in our configuration.

- Cisco Systems Cisco 871 Integrated Services Router
- Broadcom NetXtreme Gigabit Ethernet network cards
- Four Intel Pentium 4 – 3 GHz machines running Windows XP Professional, Service Pack Three, with 1GB of RAM.

To run a real-world test and in order to demonstrate the key functionality of our VCUDSDC framework, we have setup several environmental parameters, which are listed below in Table 6.1 Appendix B.

All the machines used within the prototype test-bed were connected using the standard TCP/IP protocol. The 1.6 version of the Java Development Kit was used on all machines within the network. Several decisions were made regarding this network configuration. The first decision being that all devices must be connected using a LAN communication. The second decision was that the 100MB standard should be used to enable multimedia streams to be processed more efficiently. The third decision was to enable devices to join and leave the network without having to inform any third party – this was designed to allow any device at any time to join or leave the network using ad hoc networking principles.

6.6.3 System Operation

To test the operational capabilities, all systems implement the virtual lab and publish all the services that contain the VCUDSDC framework. Each object in the Virtual Lab also publishes the application specific services it provides. For example, the SunSpot object publishes Temperature, Accelerometer and Light behaviour services. Objects that require behaviour begin by trying to discover behaviour based on the behavioural Meta data they require. For example, the SunSpot object begins by trying to find behaviour that can get temperature, accelerometer and light information in Virtual and Physical worlds. Once objects have published and run all functions they remain in an inactive state.

Using the user interface, we tested whether our prototype could dynamically generate behaviour and its visual effect and facilitate sharing digital content. For example, we tested the discovery of SunSpot objects by manipulating the details described in the object capability model, *i.e.* specified that objects must have temperature, accelerometer and light information capabilities. Our implementation illustrated that this could be effectively achieved.

When the virtual lab starts up it loads all object one by one, and load its behaviours and visual effects from a data source. If behaviour is not found then using rules and the scripting engine with Ontologies it starts finding behaviours for an object. For example, a system that discovers SunSpot objects to get temperature, accelerometer and light behaviours. Using the quality of service features supported within VCUDSDC, our framework was capable of selecting the best possible behaviour for each object within our network configuration. The prototype also demonstrated that when a better behaviour is found it could successfully update the script to improve the functionality. Overall, the operational functionality exhibited by our prototype illustrates that secondary and application specific services could be seamlessly integrated and removed from the network without disrupting service compositions.

6.7 Summary

This chapter has described the main implementation details used to evaluate our VCUDSDC framework. We have used different tools to implement our framework, which includes graphic to networking systems. In this chapter, we have implemented various components of our proposed prototype; we have used Blender to design the 3D model for our virtual lab and imported it to Java for rendering purposes.

We have implemented a Drools rule-base system to generate behaviour dynamically along with a Java scripting engine to compile and run scripts. P2P communication is implemented using SUF, which is based on JXTA technology to allow us to discover and communicate with other peers over the network. The purpose of our implementation chapter was to demonstrate an idea and make sure that the requirements and challenges presented in Chapter 1 could be addressed.

Chapter 7

7 Evaluation

7.1 Introduction

The previous chapter described the implementation of our framework and in this chapter, we present the evaluation of our work. The requirements we have presented in chapter 1 need to address some of the limitations with existing networked appliances and home networking approaches. These requirements detail what is necessary to allow flexible appliances and middleware solutions that will enable virtual environments to extend the functionality of physical devices in home networked appliances and automatically generate functionality without human intervention. Each of these requirements outlines the foundation for the qualitative evaluation of our proposed framework.

7.2 VCUDSDC Framework

The key requirement was to provide an open middleware architecture to visualise networked appliances in virtual worlds and facilitate sharing of digital content across heterogeneous environments (both physical and virtual). Using high-level semantics and rules to discover or generate behaviours automatically and to make it easy for end users to control physical devices and emulate the functionality from virtual worlds and to utilize the hidden services in Home networks to extend physical environments. The idea is based on current file sharing principles whereby popular files are distributed, shared and discovered within a P2P network. Our approach goes further than simple file sharing. We can share objects and its behaviour across different environments and even generate behaviour in the environments where that behaviour does not exist [Shaheed 2007]. This makes our framework robust and highly flexible and our framework ensures objects use behaviour that has been discovered within the network provided by its original environment or other peer environments.

One of our key framework requirements is flexibility and to allow digital content to be shared across different heterogeneous environments. The framework allows devices to join the network and leave it at any time. To generate behaviour dynamically Meta descriptions attached to objects enable environments to discover or generate behaviours on the fly. For example, if the object is a vehicle, the system will break its functionality into different parts and find the behaviour for each part individually. These are combined into one script and attached to that object.

Our framework adopted the same approach as SUF [Fergus 2005] to discover behaviours which are abstracted from functions objects provide. Object capability-matching algorithms that use high-level semantics and metadata allows our system to reason over that behaviour it provides using rules which is demonstrated in our framework implementation. Behaviours are generated using high-level semantic descriptions and mapped to objects, which closely match that description. The framework broadcasts object behaviour requests enclosing the semantics that define the required behaviours.

Furthermore, we have extended the concepts surrounding P2P, whereby we not only focus on simple file sharing but also allow users to share active entities such as a weapon in a game. P2P is typically associated with file-sharing, however these overlay networks can offer much more by sharing objects and their behaviours. We have made a number of novel contributions within this area and demonstrated how virtual environments can be used to enhance and extend networked appliances and home networking capabilities [Shaheed 2008; Fergus 2008; Shaheed 2008; Amjad Shaheed 2009]. To our knowledge, our framework is the first to use dynamic behaviour generation with shared objects across different heterogeneous environments (both physical and real). It disperses devices and hidden services within networked appliances and allows control and operation from virtual environments, which are not possible using physical controls. We have demonstrated that this approach is achievable using our prototype, which has revealed that key functions, illustrated in this thesis and which are not presented by other approaches such as Second Life, World of Warcraft, can be realised.

7.3 Visual Engine

A further key requirement addressed in our framework is a visual container where we can effectively display devices and virtual objects, such as gaming characters in computer-simulated environments called virtual worlds. Our framework uses an open source Java gaming engine called JME (JavaMonkey engine) to render objects in virtual environments, which we have demonstrated in our implementation chapter. JME allows us to import graphical models and load them into the Gaming Engine, thus it makes it easy to design 3D models in other application such the Blender or 3D Max and then save it to models in physical devices such as sensors, mobile phones, TVs, Fans and lamps. When they are connected to the virtual world, they send visual models to virtual environments for rendering. We go one-step further and allow virtual environment to discover or draw basic models if a device cannot provide its visual information using the semantic description of an object.

Our framework ensures the correct 3D model has been loaded if supplied by an object and checks the structure of the 3D model, if the 3D model is not supported then it will propagate a message in the network to find a suitable 3D model for that object. If the result does not arrive in specific time then the system will allocate basic 3D models unless it finds the right one in a later stage. This makes the framework more flexible [Shaheed 2007].

7.4 Asset Lookup and Resource Monitor

There is a need to manage resources in the framework, i.e. what devices are connected to the virtual environment, its capabilities, and the probability of it staying connected. Devices can come and go at any time and there is no restriction. For example, a wireless sensor device transmitting temperature or light data as discussed in the case study calculates the battery status and keeps the information in the asset lookup table. This allows users to know how much time they have to utilize the device functionality. And furthermore, if a SunSpot has shared its object with other peers

then it should keep track of how much battery life has been consumed [Shaheed 2007].

7.5 Rule Engine

We discussed the Rule-Base system in detail in Chapter 2 section 3. Our framework provides a dynamic environment where objects can join and its functionality generated on fly. To achieve this we have used a Rule-Based System called Drools to write simple rules and combined it with a database system to access information, and to reduce rule-processing costs. Although performance is not our main issue behaviours are not created for those objects, which have already been discovered, each time the virtual environment is started.

We have implemented Drools, which is easy to use to write simple or complex rules in a graphical interface. A script can be created firing one rule or a set of rules and each rule can create multiple scripts. For example, a table Lamp can have multiple rules to generate different behaviour like switch On, Off and flash light behaviours [Shaheed 2008].

7.6 Scripting Engine

In dynamic environments where objects can be added at run time and functionality needs to be created on the fly, scripting languages provide capabilities to insert a piece of code into a running application to form part of that application. The requirement is to support multiple scripting languages to support flexibility. This allows JavaScript ecmascriptand Linden Script amongst other to be supported.

To enhance the functionality of our framework we have implemented the Java scripting engine, which supports multiple scripting languages as was explained in the Implementation Chapter. It was more important in our framework to have this functionality in order to share objects across heterogeneous virtual environments and to provide choice with different scripting languages.

We have evaluated the scripting language component by introducing scripting evaluation functions to check the syntax of scripting languages before executing and load appropriate scripting engines. We go back to our Case Study SunSpot example where JavaScript was used to create functionality for the SunSpot in our virtual lab and to demonstrate how we can create functionary on the fly using rules, scripting and semantics [Shaheed 2008].

7.7 Object Behaviour Matching

As more devices and virtual objects that join the virtual environment it is important to select the best functionality. Our framework supports this requirement in order to make sure the best behaviour is selected for a particular object. It is necessary to describe behaviour parameters either by device manufactures if it is physical devices or the developer of virtual objects to assess that functionality the object must have including their associated functionality [Shaheed 2008].

7.8 Comparison with Existing Approaches

In this section, we compare our framework with three existing networked appliance, virtual environment and Home Networking approaches. We use our novel contributions as a basis for our comparison and compare them to the corresponding features provide by these architecture such as RUNE, Second Life and World of Warcraft.

7.8.1 RUNES

The RUNES project provides a framework that allows any device to form part of the Internet, irrespective of its capabilities. RUNES is a services-oriented architecture which allows services to be integrated within the network and discovered. RUNES provides a solution but like USB it ties device manufactures into their protocol. In our framework, we have avoided this limitation by using ontological structures with rules to reason about what functionalities objects provide and how they can be combined. In our framework if a behaviour is not found locally then it propagates behaviour requests that describes the behaviour required. This is achieves using a novel approach based on semantic web technologies.

RUNES does not provide mechanisms to create functionality on the fly based on how capable they are or how effectively they can execute the behaviours they provide. In order to allow true ubiquity it is necessary to have mechanisms to allow virtual environments to decide what behaviour they support in order to create behaviour dynamically. We have addressed this limitation using reasoning services for description and composition. In our framework the object functionality can be formulated using rich ontological data propagated as semantic requests within the network, which can be matched against the semantic description of an object. This makes our behaviour generation far more flexible, scalable and less restrictive. Using visualization of devices in virtual environments also provides more control over physical devices and can increase device capabilities, which is not possible in the real world.

7.8.2 Second Life

Second life is a 3D simulated environment where people interact using avatars. Users can create digital content and trade objects for money. Second Life allows users to create digital content and functionality and share them with other Second Life users. The limitation of Second life is, it does not allow people to share content across other environments such as Sim Online. Our framework allows users to share content across heterogeneous virtual environments i.e. Sim Online users can share content with Second Life users and vice versa.

To create functionality for objects in Second Life users have to use its scripting language. However, the content is fixed and no other is allowed. Our framework supports multi scripting and allows users to choose the scripting language they prefer. Second Life does not allow behaviours to be created on the fly using ontologies and rules. Users have to write a script manually and attach it to that object. Our approach searches for behaviours and dynamically attaches behaviours to an object without human interaction.

Second Life is a Client-Server architecture, which make it more unreliable if a server fails then the whole service become unavailable. Our framework utilities P2P technologies to support such failures.

7.8.3 World of Warcraft

World of Warcraft is a massively multiplayer game played online where users can trade gaming object with other users. World of Warcraft does not allow users to create avatars but they are allowed to customize the character and its skill from time to time. It allows users to share its contents within World of Warcraft like our framework. However, it does not allow content to be shared across different environments. For example, a Second Life avatar cannot be imported into World of Warcraft, which is the case in our framework.

World of Warcraft does not allow functionality for objects to be developed on the fly without human interaction nor does it allow users to write scripts. This makes the World of Warcraft inflexible.

And lastly, World of Warcraft does not have functionality to support multi scripts or dynamic generation of behaviour on fly.

7.9 Framework Test and Evaluation

This section provides a test and evaluation for the VCUDSDC architecture in this thesis. Small wireless sensors devices and bigger devices like a Media player, TV, Lamp and Fan are shared between two virtual environments. The criteria set in this section are to generate object behaviour on the fly for both physical and virtual objects and share objects between two virtual environments.. Our initial results have been published [Shaheed 2010] to demonstrate the working prototype's.

We have experimented with different devices to evaluate our framework in order to ensure that we can successfully create behaviour at run time and visualise them in virtual world we have setup few experiments discussed in following sections..

7.9.1 Evaluation of Framework to generate behaviour for objects.

We have setup two virtual environments on two different computers that are connected using a Local Area Network with speeds up to 10MB/Sec. The computer specification is a P4 with a 3Gz Intel process and 1GB RAM. We have add a new lamp object to Computer A. The system detects the lamp object as shown in the Log file (line 443 in Appendix C) and starts loading metadata from the lamp: both meta description and its 3D model. It took 4 seconds to transfer both the metadata and the 3D model (the 3D model contains 2MB of data and 1KB of meta description for the Lamp) from the lamp to computer A. The system searched the object functionality locally but could not found it. Consequently it sent a request over the network (as shown in Appendix C line 451). The implementation is discussed in Chapter 6. The Log File also shows that it took four seconds to receive a response over the network from Computer B (Appendix C line 450), which has the behaviour for the Lamp and after accepting the behaviour from Computer B, it took 4 seconds to transfer the behaviour from Computer B to Computer A (as shown in Appendix C line 457). Once Computer A received the script from Computer A for the Lamp it started validating the script to make sure it does not contain any errors. The Log file shows there is no syntax error in the script (Appendix C line 461), so the system attached the script to the Lamp object and started rendering the object in the virtual world. This took around another four seconds. The whole process took 16 seconds to render the 3D model and search its behaviour over the network as shown in Table 7.1.

Description	Time Taken (sec)
Meta Data Load to Virtual World	4
Behaviour Result Locally	1
Searching on Network Got first response	4
Transfer Behaviour	1
Validating Behaviour	1
Rendering Object	5

Table 7.1 Behaviour generation and rendering for Lamp object introduced for the first time

We have repeated this process three times for the Lamp object as shown in Table 7.2 where the average time to add the Lamp to Computer A was 15.8 seconds including finding the behaviour over network and transferring this to Computer A.

Attempts	Time Taken (sec)
First Time	16
Second Time	16.5
Third Time	15
Average Time	**15.83**

Table 7.2 Numbers of times the Lamp was added to the Virtual World

The full log files for all three experiments are provided in Appendix C. The same virtual environments were used to transfer a Fan object, which has a 4MB 3D model and a 10KB script file. This took on average 2 seconds to transfer the Fan object from one virtual environment to the other virtual environment and took a further 10 seconds to render the Fan object. The result was an average time jof 17 seconds for the whole process as shown in Figure 7.1.

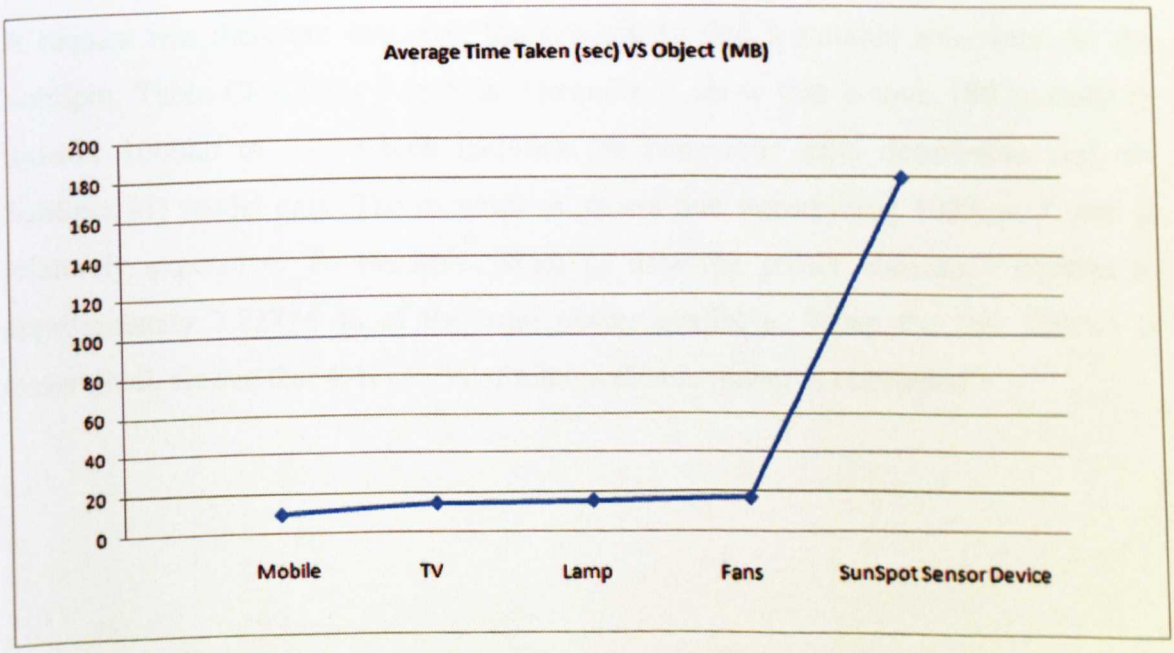


Figure 7.1 Time Taken (seconds) vs. Object Creation

Object	Average Time Taken (sec)
Lamp	15.83
Fans	17
Mobile Phone	10
TV	15
SunSpot Sensor Device	180

Table 7.3 Average Time Taken for complete process for different objects

7.9.2 SunSpot to Virtual Environment

In our second experiment, we connected a SunSpot device to a virtual world to test the framework on smaller devices. Using two criteria, we evaluated our framework. First, to generate behaviour on the fly using a meta data description stored on the SunSpot devices and second, to visualize the device functionality and control it from the virtual word.

In our first experiment, we tested our SunSPOT multi-hop network using 4 nodes. Data (1000kb) was requested from the virtual environment via the base station. The framework detected the SunSpot and started transferring the meta data and 3D model from the device. Once the meta description had been retrieved the framework searched the system in its entirety, but the behaviour in local caches could not found. A request was therefore sent over the network to find a suitable behaviour for the SunSpot. Table C8.4 lines 3 to 5 in Appendix C show that it took 180 seconds to transfer 1000kb of data which includes the behaviour meta description and the SunSpot 3D model data. The experiment shows that transmitting 1000kb of data is relatively expensive. To transmit 500kb of data the power consumed equates to approximately 2.22754 % of the total power available. When the full 1000kb is transmitted, we see that 4.47135 % of total available power is consumed.

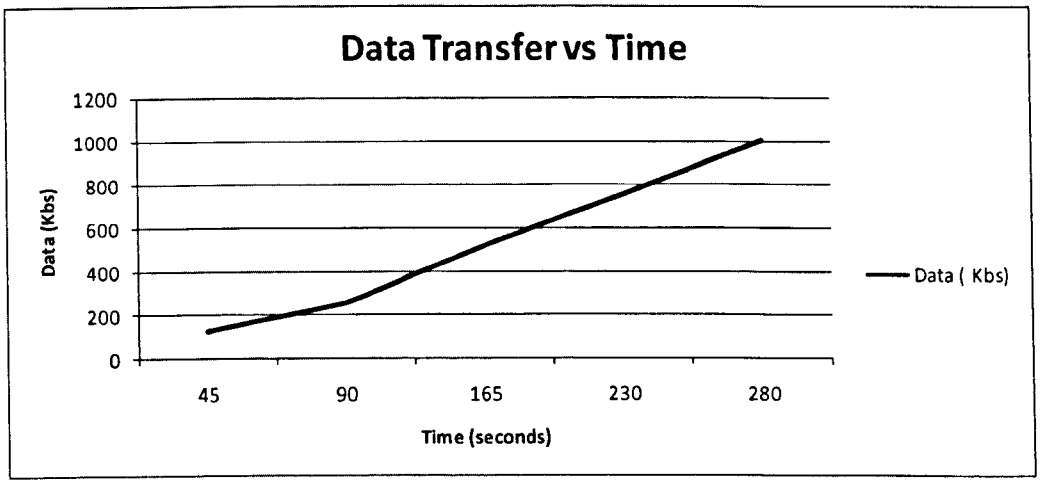


Figure 7.2 Data transfer versus time.

The overall results provide a guide for data and power consumption. Based on application requirements, user acceptance tests and sensor configurations are needed. This said we anticipate far quicker times through better data compression and increases in throughput as advances are made. What the results do show is that devices, despite their size, have the ability to describe and present themselves for usage at any level of detail, i.e. to allow magnifying of its physical attributes.

7.10 Summary

Our framework has achieved the objectives as expected and it has demonstrated that the challenges highlighted in Chapter 1 have been addressed.

Our evaluation shows that our framework exceeds current research innovation within networked appliances; home networking and computer simulated environments such as 3D gaming and addresses a number of problems. We have provided results ranging from small sensor devices to more powerful computational devices such as a computers to demonstrate the working prototypes and their overall performance. The virtual environments are processor hungry and getting visual information from small sensor devices can affect the performance of such devices but we have demonstrated that even small devices can perform sufficiently.

Chapter 8

8 Conclusion and Future Work

8.1 Introduction

In this thesis we have stated that the large number of home appliances and the complex functions they provide make it ever harder for a specialist, let alone a regular home user, to write functionality for these devices and use them. The focus of this project was to provide solutions where objects can be included in a virtual environment (both physical and virtual) and their functionality created on the fly using rules and ontologies to automate the process of generating behaviours dynamically. This was realised in the design of the Ubiquitous Home Devices and Service architecture (VCUDSDC) and its functional components. It enabled networked home appliances to be dynamically integrated and visualised in a virtual environment to provide implicit solutions while integrating the entire network in a heterogeneous manner.

This final chapter presents a summary of the thesis, its results and contributions. From these, we identify future guidelines and other issues related with the enabled environment. The chapter is sealed by concluding remarks.

8.2 Thesis Summary

Chapter 1 of this thesis provided an overview of the problem domain, namely the inefficiencies associated with current networked appliances and home networking and virtual environments such as games and social networks. It identified that little work has been carried out within ad hoc home network environments, which take into account flexible mechanisms that enable devices and the services they provide to be automatically integrated and where new objects can be supported through visualisation and dynamic behavours. This has moved us towards a true automated

software development platform in terms of functionality generation. This chapter then briefly detailed a framework we developed that addresses these limitations enabling objects (both real and physical devices) to be automatically integrated using flexible algorithms that perform the integration process. This chapter concluded that by defining the scope of the research project, the novel contributions where made.

In Chapter 2 we presented the background work. The aim was to provide a better understanding of the need to share digital content across different environments and explain their purpose of operation starting from its history towards various types of virtual environments such as social 3D virtual environments and games. Chapter 2 described dynamic scripting languages and its purpose. Then looked at rule-base systems and highlighted their different methods for processing rules such as Backward Chaining and Forward Chaining. This chapter also explained Peer-to-Peer networking and its various models.

In Chapter 3 Ubiquitous/Pervasive computing and Networked Appliances where discussed and the challenges presented. Furthermore, networked appliances and home networking, including current middleware solutions where discussed that aim to seamlessly interconnect devices within home networks.

In chapter 4 we described the UML design models for all the components that comprise our framework. These components allow virtual environments to share objects and form functionality on the fly. Using the framework the behaviour can be discovered using high-level semantics.

In chapter 5 a Case Study has been presented to give a real world scenario to show a possible implementation for our framework. The Case Study focuses on how objects can be visualised and how behaviour can be developed on the fly and integrated in virtual worlds. We use a Smart Home Network as an example how we can control physical devices and automate the integration process with in virtual worlds.

In Chapter 6 we discussed the implementation of our framework. It described the issues that arose during the implementation of the framework components as per the

requirements and design laid out in Chapter 4. We used advances in P2P technologies, which is devoid of any centralisation and the Java object-oriented programming language to implement our prototype. This chapter presented a set of test scenarios we used to test the prototype within the virtual environment setup of our own laboratory. These scenarios assisted in system integration testing and the evaluation. The prototype successfully demonstrates the applicability of our framework and acts as a strong proof-of-concept for our solution.

An evaluation of the system was presented in Chapter 7, which compares and contrasts our approach with other relevant work in this field. In the chapter we have evaluated the performance of our system in terms of processing and transferring objects between heterogeneous environments such as how long it takes to transfer objects from one virtual world to another and vice versa. Testing of our framework is carried out in this chapter to show how flexible our framework is.

8.3 Contribution to knowledge

In this section, we present our contributions to knowledge with the development of our framework for visualizing and controlling Networked appliances and to facilitate sharing digital content across different virtual environments. The challenges we have overcome in order to achieve this include: sharing objects (both real and virtual), dynamic generation of behaviours, behaviour discovery using high level semantics, visualization of networked appliances in virtual environments to enhance the functionality of physical devices, and to control devices from within virtual environments. We have addressed these challenges using our framework and made several novel contributions [Shaheed 2007], [Fergus 2008], [Fergus 2008], [Merabti 2008b], [Shaheed 2008], [Shaheed 2009], [Shaheed 2009], [Haroon 2009], [Shaheed 2010], [Haroon 2010]. In the following sub section, we discuss our novel contributions.

8.3.1 3D Visualization in Ubiquitous Computing, Device and Services

In the area of service-oriented networking we have made several novel contributions, which we have published in [Shaheed 2010], [Shaheed 2009], [Haroon 2009] – each contribution is listed below:

- Our framework allows application functionality to be embedded within the environment as network services. The operational functions provided by devices are dispersed within the network as services that can be combined to create high-level applications and these services can be provided and exploited by any type of network device.
- Our framework allows devices to be visualised in virtual environments in order to extract and use device functionality and hidden services within the network in a unique and novel way to provide greater control.

8.3.2 Content Sharing across different Environments (Virtual and Physical)

We have made several contributions in the area of digital content sharing, which we have published in [Shaheed 2007], [Merabti 2008b]. These contributions are listed below:

- Our framework facilitates the sharing of real time objects across different Environments (Both real and virtual). This allows us to move objects from one environment to another. For example, a weapon in a game can be shared between World Of Warcraft and Halo and vice versa.
- It is difficult to transfer objects between very different environments and even more difficult to generate behaviours on the fly. Our framework provides mechanisms to generate behaviours on the fly 'intelligently; process the signatures they provide that allows the environment to understand how the object operates and how it can be integrated [Shaheed 2008].

8.3.3 Object Behaviour Matching

In the area of Behaviour Matching we have made several novel contributions, which we have published in [Shaheed 2008] which are listed below:

- Our framework utilises semantic annotations with rules to create dynamic environments and generate scripts on the fly, and then project them into heterogeneous virtual environment in a novel way. Once behaviour is matched, it is assigned to the object. Object along with their behaviours are stored in the system for later use to reduce overheads on the system.

8.3.4 Interaction and Control in Ubiquitous Computing System

In Ubiquitous Computing, we have demonstrated that our framework can visualize small devices and hidden services in virtual environments to enhance its limited capabilities and allow interaction and control. We have made several novel Contributions, which again we have published in [Shaheed 2010], [Haroon 2010].

- We have developed our framework to discover, visualize and control devices in Ubiquitous Computing Environments. By equipping devices with visualization and behaviour data they are able to describe themselves by providing this data to other services. When a device is discovered its behaviour is generated on the fly and interaction can be achieved via virtual environments.
- Virtual environments provide full control over device operation and make it easier to operate virtually than physically because of their associated limitations.
- Our framework allows us to simulate the functionality of devices in virtual environments to perform experimentation instead without having to physically use the device or be in the same location [Shaheed 2009].

8.4 Future Work

So far, in this chapter we have run through the project aims, main findings and results, and considered the novel contributions of our work. However, our work has also come across difficulties and has raised a number of interesting questions. This section deals with, in our view, the most significant of these questions.

8.4.1 Visual Engine

In this research, we used predefined models to render 3D objects, however the challenge is to generate 3D models on the fly from photographs. This would make the approach more dynamic and flexible in terms of visualisation and operation.

In our future work we also need to consider the performance in more detail and compare this with other approaches, such as Web-based gaming. Whilst performances may be problematic, it is not envisaged to be a significant problem in the future given that better communication mediums are available, i.e. 8MB and 16MB broadband connections are now commonplace. The files containing the 3D metadata are currently around 5MB, which can be problematic as the number of objects downloaded to the container increases. This needs further work and evaluation. Perhaps a look at compression techniques would help.

8.4.2 Rule-Engine and Ontologies

Whilst we have presented an initial prototype system, it is clear that much work remains to be carried out before a fully effective system is produced. In particular, working with rules in conformance with ontologies needs to be understood because ontology processing is challenging and time consuming. We hope to extend the use of ontologies in the system in order to increase the robustness of interactions between components, allowing for greater flexibility in the way components represent themselves.

Early results have shown that using rules provides a powerful way to dynamically create scripted behaviour from semantic descriptions. We have implemented a few rules to test our prototypes but the biggest problem is designing the correct set of rules that leads to the behaviour we want. This will require enormous ingenuity to think of the right set of rules and the laborious process of generating the rules will often have to be repeated.

8.4.3 Security

In our research, we have not implemented security or considered it as a main research area except for basic authentication provided by the JXTA framework. Because of the Ad hoc nature of our environment, it is possible that content received from services is intercepted and altered during transmission and it is possible that people will transfer dangerous behaviour to crash the whole system or take control of it.

In the future, lightweight trust mechanisms need to be developed which guarantee that the data transferred between services has not been altered or redirected during transmission. This mechanism must accommodate different levels of integrity dependent on what type of data is being transferred. Development will ensure that the computational overhead incurred by the posited scheme is minimal, whilst ensuring that the content received by virtual environments is free from modification.

Better scripting engines need to be developed in order to validate scripts and check any harmful code, which could potentially be dangerous for the system and result in crashing or the stealing of objects without the knowledge of the owner.

8.4.4 Script Parser and Generator

This implementation specific module structures the script according to the scripting language being used. We have implemented and tested the Java Scripting Engine, which covers most common scripting syntax. However, other powerful scripting tools will be introduced thus we need to upgrade the scripting engine. The framework needs to be abstracted to represent efficiently many of the complexities it currently supports. For example, more flexible mechanisms that enable interoperability between different scripting languages including rule engines, are needed.

8.5 Concluding Remarks

The home is becoming more embedded with wireless technologies and digital services. For example, machine-to-machine solutions for utility billing and

management are now possible using cellular networks. With many homes now supporting different communication capabilities it has become relatively easy to deploy ubiquitous devices (sensors) and services for any number of different applications.

This is already practical where probing the home network often uncovers services not necessarily known beforehand, e.g. universal plug-n-play devices like cable modems provide passive interactive services. Invisible services like these are set to accelerate as more networked devices form part of the home itself. Consequently, there will be a need to visualise available devices or services in order to allow users to better exploit the functions they provide.

In this thesis, these requirements have been successfully addressed by designing and developing a new framework called VCUDSDC. This framework allows us to visualise devices and hidden services, providing better exploitation of the services they provide. Our framework allows us to manipulate physical devices in new and novel ways, i.e. increase their size to make them more visible; or combining them with other devices to provide a better means of interaction. For example, a digital magnifying glass would allow us to see what invisible technology looks like and how we may use it. We could even locate its position within the home.

The benefits of interoperability between networked devices and virtual worlds are obvious: for example, devices can be controlled remotely because of improved interaction and; secondly, both real-world devices and virtual worlds can freely share functionality and content between the two. Our framework facilitates the generation of behaviour using ontological descriptions of an object. These descriptions describe the high-level concepts that relate to the "what" part of the behaviour generation rather than the "how". As a result, each object provides ontological descriptions, which are dynamically developed over time. Once ontologies are processed, rules are used to generate scripts and to attach them to an object. Object behaviours are generated based on the low-level signatures devoid of any human intervention.

Our framework successfully incorporates varied functionalities and capabilities using mechanisms that perform behaviour matching. Before behaviours provided by objects are generated, the framework determines if the object providing the behaviour has the required hardware, software and networking competences to successfully execute it.

In this thesis, we have presented a detailed overview of the background and related work and discussed the most important factors. Developing our framework has been multi-disciplinary which has utilised and extended existing research initiatives, open standards to produce a flexible system that allows digital content to be shared, discovering, and then using them in virtual environments. Thus, this thesis provides a broad platform on which to integrate next generation networked appliances and games.

Overall, this has been a successful research project and has generated a lot of interest. It has allowed us to investigate how technological advances will progress and we believe that we are ahead of current solutions. We have successfully implemented our framework and produced a prototype that effectively demonstrates how our framework services work. The prototype implements our case study, which is a Smart home environment, and illustrates how individual functions presented by objects can be distributed within the network and used to create high-level applications. Our approach is novel, which is reflected in the number of papers we have published.

REFERENCES

[EVE Online 2006] "EVE Online", http://www.eve-online.com/, (Accessed:2006)

[Planetside 2006] "PlanetSide", http://planetside.station.sony.com/, (Accessed 2006)

[Sims Online 2006] "The Sims Online", http://www.ea.com/official/thesims/thesimsonline/, Accessed (2006)

[Star wars 2006] "Star Wars Galaxies", http://starwarsgalaxies.station.sony.com/, (Accessed 2006)

[(TME) 2003] T. M. E. (TME), "GLUE", http://www.themindelectric.com, (Accessed:2003).

[Ahearn 2007] L. Ahearn, "Terrain," in *3D Game Environments*, Focal Press, edition. Boston, (2007).

[Akyildiz 2002] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38 (4), pp. 393-422, (2002).

[Balakrishnan 2003] H. K. Balakrishnan, M. Frans; Karger, David; Morris, Robert; Stoica, Ion "Looking up data in P2P systems," *Communications of the ACM*, vol. 46 (2), pp. 43-48, (2003).

[Ballance 2008] E. J. Ballance and L. Hwajung, "Developing Proactive Information Delivery System through Bluetooth in Ubiquitous Networks", Proceedings of the *Multimedia and Ubiquitous Engineering, 2008. MUE 2008. International Conference on*, pp. 413-418, 2008)

[Barr 2007] P. Barr, J. Noble and R. Biddle, "Video game values: Human-computer interaction and games," *Interacting with Computers*, vol. 19 (2), pp. 180-195, (2007).

[Bellifemine 2008] F. Bellifemine, G. Caire, A. Poggi and G. Rimassa, "JADE: A software framework for developing multi-agent applications. Lessons learned," *Information and Software Technology*, vol. 50 (1-2), pp. 10-21, (2008).

[Berners-Lee 2001] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web," *Scientific America*, vol. 284 (5), pp. 34-43, (2001).

[Bichier 2006] M. Bichier and K. J. Lin, "Service-oriented computing," *Computer*, vol. 39 (3), pp. 99-101, (2006).

[Brian 2009] D. J. Brian, R. Claudia, W. bola and A. R. Wendy, "LIFELONG INTERACTIONS Supporting healthy aging with new technologies," *interactions*, vol. 16 (4), pp. 48-51, (2009).

[Bricon-Souf 2007] N. Bricon-Souf and C. R. Newman, "Context awareness in health care: A review," *International Journal of Medical Informatics*, vol. 76 (1), pp. 2-12, (2007).

[Captive Imagination 2007] Captive Imagination, "The open-content repository for games", http://captiveimagination.com/svn/public/cigame/trunk/, (Accessed:2007).

[Castro 2002] M. D. Castro, P.; Kermarrec, A.-M.; Rowstron, A.I.T, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20 (8), pp. 1489-1499, (2002).

[Chawathe 2003] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker, "Making Gnutella-like P2P systems scalable", Proceedings of the

ACM SIGCOMM 2003: Conference on Computers, Karlsruhe, Germany, pp. 407-418, (25 - 29 August 2003)

[Chen 2005] R. Chen and W. Yeager, "Poblano: A Distributed Trust Model for Peer-to-Peer Networks", Sun Microsystems, http://www.jxta.org/docs/trust.pdf, pp. 1-26, (2005)

[ClassMate 2010] ClassMate, "Class Mate", http://www.classmates.com/registration/index.jsp, (Accessed:2010).

[Cory 1999] D. K. Cory, O. Robert, D. A. Gregory, G. A. Christopher, A. E. Irfan, M. Blair, D. M. Elizabeth, S. Thad and N. Wendy, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research," in Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture, vol., Ed.^Eds., ed.: Springer-Verlag, 1999, pp.

[Costa 2005] P. Costa, G. Coulson, C. Mascolo, G. P. Picco and S. Zachariadis, "The RUNES middleware: a reconfigurable component-based approach to networked embedded systems", Proceedings of the *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, pp. 806-810 Vol. 2, 2005)

[Coyle 2008] C. L. Coyle and H. Vaughn, "Social networking: Communication revolution or evolution?," *Bell Labs Technical Journal*, vol. 13 (2), pp. 13-18, (2008).

[Cunningham 2001] S. Cunningham and M. J. Bailey, "Lessons from scene graphs: using scene graphs to teach hierarchical modeling," *Computers & Graphics*, vol. 25 (4), pp. 703-711, (2001).

[Clinton 2008] Y. Clinton, "Modeling a 3D Character," in Game Character Modeling and Animation with 3ds Max, Focal Press, edition. Boston, (2008).

[D'Aubeterre 2009] F. D'Aubeterre, L. S. Iyer, R. Ehrhardt and R. Singh, "Discovery process in a B2B eMarketplace: A semantic matchmaking approach," *International Journal of Intelligent Information Technologies*, vol. 5 (4), pp. 16-40, (2009).

[D-LINK 2010] D-LINK, "D-Link", http://www.dlink.com/products/?pid=438&sec=0, (Accessed:2010).

[DeVoss 2006] D. N. DeVoss and J. E. Porter, "Why Napster matters to writing: Filesharing as a new ethic of digital delivery," *Computer and Composition*, vol. 23 (2), pp. 178-210, (2006).

[Dimitrov 2007] D. Dimitrov, K. Schreve, A. Taylor and B. Vincent, "Rapid prototyping driven design and realisation of large components," *Rapid Prototyping Journal* vol. 13 (2), pp. 85 - 91, (2007).

[DLNA 2004a] DLNA, "DLNA: Overview and Vision", http://www.dlna.org/about/DLNA_Overview.pdf, (Accessed:2006).

[DLNA 2004b] DLNA, "DLNA: Overview and Vision", http://www.dlna.org/about_us/roadmap/DLNA_Whitepaper.pdf, (Accessed:2010).

[Drools 20011] Drools, "Drools-The Rule Engine", http://downloads.jboss.com/drools/docs/4.0.7.19894.GA/html/index.html, (Accessed:2009).

[Druschel 2001] P. R. Druschel, A, "PAST: a large-scale, persistent peer-to-peer storage utility", Proceedings of the *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, Schoss Elmau, Germany, pp. 75-80, (20-22 May 2001)

[Ducheneaut 2006] N. Ducheneaut, N. Yee, E. Nickell and R. Moore, "Building an MMO with Mass Appeal: A look at Gameplay in World of Warcraft," *Games and Culture - Journal of Interactive Media*, vol. 1 (4), pp. 281 - 317, (2006).

[Eberly 2007] D. H. Eberly, "Special Effects Using Shaders," in *3D Game Engine Design (Second Edition)*, Morgan Kaufmann, edition. San Francisco, (2007).

[El-Nasr 2006] M. S. El-Nasr and B. K. Smith, "Learning Through Game Modding," *ACM Computers in Entertainment*, vol. 4 (1), pp., (2006).

[Engine 2010] j. Engine, "Java Monkey Engine User Guide", www.jmonkeyengine.com, (Accessed:2010).

[Erwin 2007] C. Erwin and V. Keith, "COLLADA physics," in Proceedings of the twelfth international conference on 3D web technology, vol., Ed.^Eds., ed. Perugia, Italy: ACM, 2007, pp.

[*Fehn* 2006] *C. Fehn*, *R. D. Barre* and *S. Pastoor*, "Interactive 3-DTV - Concepts and Key Technologies," *Proceedings of the IEEE*, vol. 95 (3), pp. 524 - 538, (2006).

[Fergus 2009] P. Fergus, K. Kafiyat, M. Merabti, A. Taleb-bendiab and A. El Rhalibi, "Remote Phsiotherapy Treatments using Wireless Body Sensor Networks", Proceedings of the *The 5th International Wireless Communications and Mobile Computing Conference*, Leipzig, Germany, pp., (21st - 24th June 2009)

[Fergus 2008a] P. Fergus, D. Lewellyn-Jones, **A. Shaheed**, M. Merabti and A. El Rhalibi, "User Controlled 3D Multimedia using Networked Appliances", Proceedings of the *5th Annual IEEE Consumer Communications & Networking Conference: Workshop on Digital Entertainment, Networked Virtual Environments, and Creative Technology*, Las Vegas, USA, pp., (10th - 12th January 2008)

[Fergus 2007] P. Fergus, D. Llewellyn-Jones, M. Merabti and A. El Rhalibi, "Bridging the Gap between Networked Appliances and Virtual Worlds", Proceedings of the *Advanced Information Networking and Applications Workshops*, pp. 935-940, (21-23 May 2007 2007)

[Fergus 2008b] P. Fergus, M. Merabti, O. Abuelma'atti and **A. Shaheed**, "Next Generation Mobile Multimedia," in *Handbook of Research on Mobile Multimedia*2nd Edition edition. (2008).

[Fergus 2005] P. Fergus, M. Merabti, M. B. Hanneghan, A. Taleb-Bendiab and A. Minghwan, "A Semantic Framework for Self-Adaptive Networked Appliances", Proceedings of the *(CCNC'05) IEEE Consumer Communications & Networking Conference*, Las Vegas, Nevada, USA, pp. 229-234, (January 2005 2005)

[Fergus 2003a] P. Fergus, A. Mingkhwan, M. Merabti and M. Hanneghan, "Capturing Tacit Knowledge in P2P Networks", Proceedings of the *(PGNET'2003)The 4th EPSRC Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool, UK, pp. 159-165, (June 2003 2003)

[Fergus 2003b] P. Fergus, A. Mingkhwan, M. Merabti and M. Hanneghan, "Distributed Emergent Semantics in P2P Networks", Proceedings of the *(IKS'2003) Information and Knowledge Sharing*, Scottsdale, Arizona, USA, pp. 75-82, (November 2003 2003)

[Fergus 2003c] P. Fergus, A. Mingkhwan, M. Merabti and M. Hanneghan, "DiSUS: Mobile Ad Hoc Network Unstructured Services", Proceedings of the

(PWC'2003) Personal Wireless Communications, Venice, Italy, pp. 484-491, (September 2003 2003)

[Forgy 1982] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, vol. 19 (1), pp. 17-37, (1982).

[Franke 1995] D. Franke, "A linear state space approach to a class of discrete-event systems," *Mathematics and Computers in Simulation*, vol. 39 (5-6), pp. 499-503, (1995).

[Gauthier 2008] C. G. Gauthier and C. Grothoff, "Bootstrapping of peer-to-peer networks", Proceedings of the *International Symposium on Applications and the Internet (SAINT)*, Turku, Finland, pp. 205-208, (28th July - 1st August 2008)

[Geer 2005] D. Geer, "Users make a beeline for Zigbee Sensor Technology," *IEEE Computer*, vol. 38 (12), pp. 16-19, (2005).

[George 2008] P. George, H. Andreas and A. Kristina, "Management flight simulators: a new approach to the development of decision support systems," *WTOS*, vol. 7 (5), pp. 415-424, (2008).

[Gnutella 2008] Gnutella, "The Gnutella Protocol Specification v0.4", http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf, (Accessed:2009).

[Gnutella 2008] Gnutella, "The Gnutella Protocol Specification v0.4", http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf, pp., (2008) (Accessed:2008).

[Gong 2001] L. Gong, "JXTA: A Network Programming Environment," *IEEE Internet Computing*, vol. 5 (3), pp. 88-95, (2001).

[Haroon 2009] A. Haroon, P. Fergus, **A. Shaheed**, K. Kifayat and M.Merabti, "Healthcare Sensor Networks and Services for the Medical Home", Proceedings of the *The 10th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool, UK, pp., (22nd - 23rd June 2009 2009)

[Haroon 2010] A. Haroon, P. Fergus, **A. Shaheed** and M. Merabti, "A Wireless Home and Body Sensor Network Platform for the Early Detection of Arthritis", Proceedings of the *IEEE International Consumer Communications and Networking Conference (CCNC)* Las Vegas, Nevada, USA, pp., (9th - 12th January, 2010 2010)

[Herman 2006] H. Herman, R. J. Coombe and K. Lewis, "Your Second Life?," *Cultural Studies*, vol. 20 (2-3), pp. 184- 210, (2006).

[Hong 2009] J.-y. Hong, E.-h. Suh and S.-J. Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36 (4), pp. 8509-8522, (2009).

[HP 2004] HP, "Jena - A Semantic Web Framework for Java", http://jena.sourceforge.net, http://www.hpl.hp.com/semweb/, (Accessed:2004).

[Hsiao 2003] H.-C. Hsiao and C.-T. King, "A Tree Model for Structured Peer-to-Peer Protocols", Proceedings of the *3rd International Symposium on Cluster Computing and the Grid* Tokyo, Japan, pp. 336-343, (12-15th May 2003)

[Hsiao 2005] T. Hsiao and S. Yuan, "Practical Middleware for Massively Multiplayer Online Games," *IEEE Internet Computing*, vol. 9 (5), pp. 47 - 54, (2005).

[James 1990] D. F. James, D. Andries van, K. F. Steven and F. H. John, "Computer graphics: principles and practice (2nd ed.)", Addison-Wesley Longman Publishing Co., Inc., edition. Isbn:0-201-12110-7 (1990).

[Jennings 2006] C. Jennings and D. A. Bryan, "P2P for communications: beyond file sharing," *Business Communications Review*, vol. 36 (2), pp. 36-40, (2006).

[Jinseok 2005] S. Jinseok, G. Gwanghoon and J. K. Gerard, "Creating ubiquitous computing simulators using P-VoT," in Proceedings of the 4th international conference on Mobile and ubiquitous multimedia, vol., Ed.^Eds., ed. Christchurch, New Zealand: ACM, 2005, pp.

[JMF 2008] JMF, "Java Media Framework (JMF)", http://java.sun.com/products/java-media/jmf/, (Accessed:2008).

[JXTA 2010] JXTA, "JXTA", http://jxta.dev.java.net, (Accessed:2010).

[KaZaA 2008] KaZaA, "KaZaA", http://www.kazaa.com/us/index.htm, (Accessed:2009).

[Kirda 2009] E. Kirda, N. Jovanovic, C. Kruegel and G. Vigna, "Client-side cross-site scripting protection," *Computers & Security*, vol. In Press, Corrected Proof, (2009).

[Knutsson 2004] B. Knutsson, B. Knutsson, L. Honghui, X. Wei and B. A. H. B. Hopkins, "Peer-to-peer support for massively multiplayer games", Proceedings of the *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, pp. 107, 2004)

[Koumpis 2005] C. Koumpis, L. Hanna, M. Anderson and M. Johansson, "Wireless Industrial Control and Monitoring beyond Cable Replacement", Proceedings of the *2nd Profibus International Conference*, Warwickshire, UK, pp., (21 - 22 June 2005)

[Kucklich 2005] J. Kucklich, "Precarious Playbour: Modders and the Digital Games Industry," *International Journal on Fibreculture*, vol. 1 (5), pp., (2005).

[Kurniawan 2008] S. H. Kurniawan, "Intergenerational Learning through World of Warcraft", Proceedings of the *Digital Games and Intelligent Toys Based Education, 2008 Second IEEE International Conference on*, pp. 98-102, 2008)

[Kwon 2007] T. Kwon and S.-M. Choi, "Deriving the Virtual World from Wireless Sensor Networks for Interaction with Consumer Electronic Devices", Proceedings of the *Consumer Electronics, 2007. ICCE 2007*, pp. 1-2, 2007)

[Lai 2009] K.-C. Lai and S.-C. Kang, "Collision detection strategies for virtual construction simulation," *Automation in Construction*, vol. 18 (6), pp. 724-736, (2009).

[Lee 2007] M. S. Lee, "When social networking meets online games: the activity system of grouping in world of warcraft," in Proceedings of the 25th annual ACM international conference on Design of communication, vol., Ed.^Eds., ed. El Paso, Texas, USA: ACM, 2007, pp.

[Levy 1984] S. Levy, "Hackers: Heroes of the Computer Revolution", Anchor Press/Doubleday, edition. Isbn:0-385-19195-2 (1984).

[Liebel-Lab 2008] K. K. I. o. T.-. Liebel-Lab, "KIT Search Engine Initiative", http://sciencenet.fzk.de/, (Accessed:2009).

[Life 2010] S. Life, "Second Life", www.secondlife.com, (Accessed:2010).

[Lili 2008] S. Lili, M. Jiangjian, L. Yushun, C. Xiaochun and H. Ronghuai, "Semantic-oriented Ubiquitous Learning object model", Proceedings of the *Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on*, pp. 1-6, 2008)

[YaCy Distributed Web Search 2011] "YaCy Distributed Web Search", http://yacy.net/index.html, (Accessed:2008).

[Loui 2008] R. P. Loui, "In Praise of Scripting: Real Programming Pragmatism," *Computer*, vol. 41 (7), pp. 22-26, (2008).

[Louridas 2006] P. Louridas, "SOAP and web services," *IEEE Software*, vol. 23 (6), pp. 62-67, (2006).

[Lua 2005] K. Lua, J. Crowcroft, C. Pias, R. Sharma and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," *IEEE Communications Survey and Tutorial*, vol. 7 (2), pp. 72-93, (2005).

[Luo 2007] C. Luo, J. Sun and H. Xiong, "Monitoring and Troubleshooting in Operational IP-TV System," *IEEE Transactions on Broadcasting*, vol. Accepted for future publication, (2007).

[Ma 2003] K.-L. Ma, E. B. Lum and S. Muraki, "Recent advances in hardware-accelerated volume rendering," *Computers & Graphics*, vol. 27 (5), pp. 725-734, (2003).

[Maciol 2008] A. Maciol, "An application of rule-based tool in attributive logic for business rules modeling," *Expert Systems with Applications*, vol. 34 (3), pp. 1825-1836, (2008).

[Maedche 2003] A. Maedche, Staab, S., "Services on the Move - Towards P2P-Enabled Semantic Web Services", Proceedings of the *Proceedings of the 10th International Conference on Information Technology and Travel & Tourism*, Helsinki, pp. 124 - 133, (31 January 2003 2003)

[Malkhi 2002] D. N. Malkhi, Moni; Ratajczak, David "Viceroy: A scalable and dynamic emulation of the butterfly", Proceedings of the *21st Annual ACM Symposium on Principles of Distributed Computing (PODC'02)*, Monterey, CA, United States, pp. 183-192, (21-24 July 2002)

[Martin 2005] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan and K. Sycara, " Bringing Semantics to Web Services: The OWL-S Approach", Springer Berlin / Heidelberg, edition. Isbn:978-3-540-24328-1 (2005).

[Maymounkov 2002] P. Maymounkov and D. Mazi`eres, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric", Springer Berlin / Heidelberg, edition. MIT, Isbn:978-3-540-44179-3 (2002).

[McIlraith 2001] S. A. McIlraith, Son, T. C., Zeng, H., "Semantic Web Services," *IEEE Intelligent Systems*, vol. 16 (2), pp. 46 - 53, (2001).

[McLaughlin 2007] L. McLaughlin, "Next-Generation Entertainment: Video Goes Mobile," *IEEE Pervasive Computing*, vol. 6 (1), pp. 7 - 10, (2007).

[Merabti 2008a] M. Merabti, P. Fergus, O. Abuelma'atti, H. A. Y. H. Yu and C. A. J. C. Judice, "Managing Distributed Networked Appliances in Home Networks," *Proceedings of the IEEE*, vol. 96 (1), pp. 166-185, (2008).

[Merabti 2008b] M. Merabti, A. E. Rhalibi, **A. Shaheed,** P. Fergus and M. Price, " Virtual Environments with Content Sharing ", Proceedings of the *The 3rd International Conference on E-Learning and Games*, Nanjing, China, pp. 328-342, (25th - 27th June 2008 2008)

[Messinger 2009] P. R. Messinger, E. Stroulia, K. Lyons, M. Bone, R. H. Niu, K. Smirnov and S. Perelgut, "Virtual worlds -- past, present, and future: New directions in social computing," *Decision Support Systems*, vol. 47 (3), pp. 204-228, (2009).

[Microsoft 2004] Microsoft, "Understanding Universal Plug and Play", http://www.upnp.org/, (Accessed:2009).

[mikeg 2008] mikeg and sundararajana, "Scripting Engines and Scripting Applications", https://scripting.dev.java.net/, (Accessed:2008).

[Mingkhwan 2004] A. Mingkhwan, P. Fergus, O. Abuelma'atti and M. Merabti, "Implicit Functionality: Dynamic Services Composition for Home Networked Appliances", Proceedings of the *(ICC'2004) IEEE International Conference on Communications*, Paris, France, pp. 43-47, (June 2004 2004)

[Mingkhwan 2005] A. Mingkhwan, P. Fergus, O. Abuelma'atti, M. Merabti, B. Askwith and M. Hanneghan, "Dynamic Service Composition in Home Appliance Networks," *To appear in Multimedia Tools and Applications: A Special Issue on Advances in Consumer Communications and Networking*, (2005).

[Minoh 2001] M. Minoh and T. Kamae, "Networked Appliances and their Peer-to-Peer Architecture AMIDEN," *IEEE Communications Magazine*, vol. 39 (10), pp. 80-84, (2001).

[Mozilla.org 2007] Mozilla.org, "Rhino: JavaScript for Java", http://www.mozilla.org/rhino/, (Accessed:2007).

[Nagaraja 2006] K. Nagaraja, S. Rollins and M. Khambatti, "From the editors: Peer-to-peer community: Looking beyond the legacy of napster and gnutella," *IEEE Distributed Systems Online*, vol. 7 (3), pp. 59-65, (2006).

[Nakazawa 2000] J. Nakazawa, T. Okoshi, M. Mochizuki, Y. Tobe and H. A. T. H., "VNA: an object model for virtual network appliances", Proceedings of the *Consumer Electronics, 2000. ICCE. 2000 Digest of Technical Papers. International Conference on*, pp. 364-365, 2000)

[Nitin Desai 2003] V. V. a. S. H. Nitin Desai, "Infrastructure for Peer-to-Peer Applications in Ad-Hoc Networks," in 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), vol. 2007, Ed.^Eds., ed. Berkeley, CA, 2003, pp.

[O'Connor 2006] J. O'Connor, "Scripting Framework for Java", http://eventhorizongames.com/wiki/doku.php?id=articles:java_scripting_framework, (Accessed:2007).

[O'Mahony 2003] D. D. D. O'Mahony, "Overlay Networks: A Scalable Alternative for P2P", http://www.dynamicobjects.com/papers/w4spot.pdf, (Accessed:2008).

[OSGi 2009] OSGi, "OSGi Service Platform Core Specification", http://www.osgi.org/download/r4v42/r4.core.pdf, (Accessed:2009
).

[OSGi Alliance 2005] OSGi Alliance, "The OSGi Service Platform - Dynamic services for networked devices", http://www.osgi.org/, (Accessed:2005).

[Ott 2006] J. Ott, "Application Protocol Design Considerations for a Mobile Internet", Proceedings of the *1st ACM/IEEE International Workshop on Mobility in the evolving Internet Architecture*, San Fransico, California, USA, pp. 75 - 80, (1st December 2006)

[Ousterhout 1998] J. K. Ousterhout, "Scripting: higher level programming for the 21st Century," in IEEE Computer Magazine, vol. 31, Ed.^Eds., ed., 1998, pp. 23-30.

[Papagiannidis 2007] S. Papagiannidis, M. Bourlakis and F. Li, "Making real money in virtual worlds: MMORPGs and emerging business opportunities, challenges and ethical implications in metaverses," (2007).

[Pranav 2009] M. Pranav and M. Pattie, "SixthSense: a wearable gestural interface," in ACM SIGGRAPH ASIA 2009 Sketches, vol., Ed.^Eds., ed. Yokohama, Japan: ACM, 2009, pp.

[Proctor 2008] M. Proctor, M. Neale, P. Lin, M. Frandsen and S. G. Jr, "The Rule Engine," in, vol., Ed.^Eds., ed., 2008, pp.

[Quake 2010] Quake, "QUAKE 2 Open Source", http://www.idsoftware.com/, (Accessed:2008).

[Ratnasamy 2001] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A scalable content-addressable network", Proceedings of the *Conference. Applications, Technologies, Architectures, and Protocols for Computer Communications (ACMSIGCOMM'01)*, San Diego, CA, USA, pp. 161-172, (27-31 August 2001)

[Remi 2006] A. Remi and C. B. Mark, "Collada: Sailing the Gulf of 3d Digital Content Creation", AK Peters Ltd, edition. Isbn:1568812876 (2006).

[RTP 2008] RTP, "RTP: A Transport Protocol for Real-Time Applications", ,http://www.ietf.org/rfc/rfc1889.txt, (Accessed:2008).

[S.Networks 2008] S.Networks, "Kazaa", http://www.kazaa.com/us/index.htm, (Accessed:2008).

[Science 2006] F. C. Science, "P2P", http://ww2.cs.fsu.edu/~jungkkim/P2P.html, (Accessed:2008).

[Seay 2004] A. F. Seay, W. J. Jerome, K. S. Lee and R. E. Kraut, "Project massive: a study of online gaming communities", Proceedings of the *CHI '04 extended abstracts on Human factors in computing systems*, Vienna, Austria, pp. 1421-1424, 2004)

[Shah 2003] R. C. Shah, R. C. Shah, S. Roy, S. Jain and W. A. B. W. Brunette, "Data MULEs: modeling a three-tier architecture for sparse sensor networks Data MULEs: modeling a three-tier architecture for sparse sensor networks", Proceedings of the *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pp. 30-41, 2003)

[Shaheed 2009a] A. Shaheed, P. Fergus, K. Kifayat, O. E. Abuelma'atti and M. Merabti, "A 3D Visualiser and Controller for Networked Embedded Ubiquitous Devices", Proceedings of the *The 10th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool, UK, pp., (22nd - 23rd June 2009 2009)

[Shaheed 2009b] P. F. **Amjad Shaheed**, Abdennour El Rhalibi, Omar E. Abuelma'atti, Madjid Merabti, Marc Price, "A 3D Interactive Commerce Framework", Proceedings of the *5th IASTED European Conference on Multimedia Systems and Applications*, Cambridge, United Kingdom, pp., (13/07/2009 2009)

[Shaheed 2010] A. Shaheed, P. Fergus, Omar E. Abuelma'atti and M. Merabti, "Visualisation of Ubuiquitous Home Devices and Services", Proceedings of the *IEEE International Consumer Communications and Networking Conference (CCNC)*, Las Vegas, Nevada, USA, pp., (12/01/2010 2010)

[Shaheed 2007] A. Shaheed, P. Fergus, A. E. Rhalibi, O. Abuelma'atti and M. Merabti, "User Generated Content Sharing across different Virtual Environments," in 5th International Conference in Computer Game Design and Technology (GDTW), vol., Ed.^Eds., ed. Holiday Inn, Liverpool, UK, 2007, pp. 44 - 53.

[Shaheed 2008] A. Shaheed, P. Fergus, A. E. Rhalibi, O. E. Abuelma'atti, M. Merabti and M. Price, "A Semantic Approach to Dynamically Constructing Behaviours in Heterogenous Virtual Environments", Proceedings of the *The 9th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool, UK, pp., (23/06/2008 2008)

[Simin 2009] L. Simin and S. Jinhai, "Application of Virtual Reality Technology in the Field of Sport", Proceedings of the *Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on*, pp. 455-458, 2009)

[Son 2009] H. Son, C. Kim and K. Choi, "Rapid 3D object detection and modeling using range data from 3D range imaging camera for heavy equipment operation," *Automation in Construction*, vol. In Press, Corrected Proof, (2009).

[Sotamma 2005] O. Sotamma, "Have Fun Working with Our Product!: Critical Perspectives on Computer Game Mod Competitions", Proceedings of the *International DiGRA Conference*, Vancouver, Canada, pp., 2005)

[Steve 2007] H. Steve, I. Shahram, B. Alex, R. Alban and B. Bill, "ThinSight: versatile multi-touch sensing for thin form-factor displays," in Proceedings of the 20th annual ACM symposium on User interface software and technology, vol., Ed.^Eds., ed. Newport, Rhode Island, USA: ACM, 2007, pp.

[Takeda 2008] A. Takeda, K. Hashimoto, G. Kitagata, S. M. S. Zabir, T. Kinoshita and N. Shiratori, "A new authentication method with distributed hash table for P2P network", Proceedings of the *22nd International Conference on Advanced Information Networking and Applications (AINA'08)*, Okinawa, Japan, pp. 483-488, (25-28 March 2008)

[Tam 2002] K. K. Tam and H. L. Goh, "Session Initiation Protocol", Proceedings of the *Industrial Technology, 2002. IEEE ICIT '02. 2002 IEEE International Conference on*, pp. 1310-1314 vol.2, 2002)

[Tsung Teng 2008] C. Tsung Teng and M. Lee, "Ubiquitous Computing in Prospect: A Bibliographic Study", Proceedings of the *Ubiquitous Multimedia Computing, 2008. UMC '08. International Symposium on*, pp. 57-62, 2008)

[Tong 2004] L. Tong, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen and T. Hopkins, "Active learning to recognize multiple types of plankton", Proceedings of the *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, pp. 478-481 Vol.3, (26-26 Aug. 2004 2004)

[Sotamma 2003] Sotamma, "Computer Game Modding, Intermediality and Participatory Culture", http://old.imv.au.dk/eng/academic/pdf_files/Sotamaa.pdf, (Accessed:2006).

[UPnP 2006a] UPnP, "UPnP Technology - The Simple, Seamless Home Network", http://www.upnp.org/specs/arch/UPnP-DeviceArchitecture-v1.0-20060720.pdf, (Accessed:2009).

[UPnP 2006b] UPnP, "UPnP™ Standards", http://www.upnp.org/standardizeddcps/basic.asp, (Accessed:2009).

[Vieira Pak 2010] M. Vieira Pak and D. Castillo Brieva, "Designing and implementing a Role-Playing Game: A tool to explain factors, decision making and landscape transformation," *Environmental Modelling & Software*, vol. 25 (11), pp. 1322-1333, (2010).

[Vilei 2006] A. Vilei, G. Convertino and F. Crudo, "A New UPnP Architecture for Distributed Video Voice over IP", Proceedings of the *5th International Conference on Mobile and Ubiquitous Multimedia* Stanford, California, pp. Article 2, (4th - 6th December 2006)

[Walker 2001] L. Walker, "Uncle Sam Wants Napster!" http://www.washingtonpost.com/ac2/wp-dyn?pagename=article&node=washtech/techthursday/columns/dotcom&contentId=A59099-2001Nov7, (Accessed:2009).

[Weiser 1991] M. Weiser, "The Computer for the 21st Century," *Scientific America*, vol. 265 (3), pp. 94 - 104, (1991).

[Welsh 2009] B. C. Welsh and D. P. Farrington, "Public Area CCTV and Crime Prevention: An Updated Systematic Review and Meta-Analysis," *Justice Quarterly*1 - 31, (2009).

[Wilson 2002] B. J. Wilson, "JXTA", New Riders Publishing, First edition edition. 201 West 103rd Street, Indianapolis, Indiana 46290, Isbn:0734712344 (2002).

[Wu 2007] J. Wu and J. Dong, "Simple service discovery and configuration protocol for embedded devices", Proceedings of the *International Conference on Communication Technology (ICCT '06)*, Guilin, China, pp. 201-204, (27-30 November 2007)

[Wu 2008] S.-y. Wu, C.-S. Chang, S.-H. Ho and H.-S. Chao, "Rule-based intelligent adaptation in mobile information systems," *Expert Systems with Applications*, vol. 34 (2), pp. 1078-1092, (2008).

[Zhang 2008] S. Zhang, H. J. and H. Zhou, "An interactive Internet-based system for tracking upper limb motion in home-based rehabilitiation," *Medical and Biological Engineering and Computing*, vol. 46 (3), pp. 241-249, (2008).

[Zhou 2008] H. Zhou, T. Stone, H. Hu and N. Harris, "Use of multiple wearable inerail sensors in upper limb motion tracking," *Medical Engineering & Physics*, vol. 30 (1), pp. 123-133, (2008).

[Xbox Kinect 2011] Xbox Kinect, "http://www.xbox.com/en-GB/kinect" (Accessed:2011)

[Nintendo Wii 2011] Nintendo Wii, " http://www.nintendo.com/wii" (Accessed 2011)[Play Station 3 2011] Play Station 3 " http://uk.playstation.com/ps3/" (Accessed:2011)

[The Blender Python 2011] The Blender Python" http://www.blender.org/documentation/244PythonDoc/index.html" (Accessed 2011)

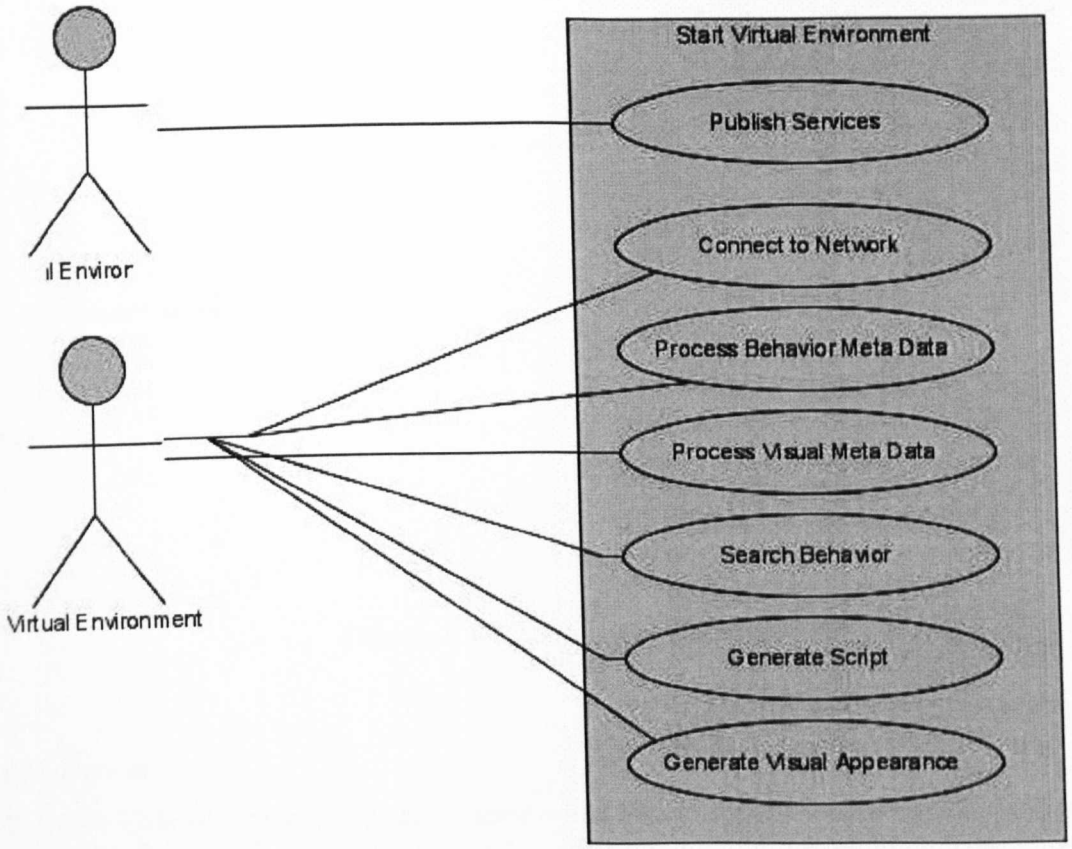


Figure A1.8.1 Start Virtual Environment

Description:

This case study illustrates a scenario when a virtual environment initially started on within this framework.

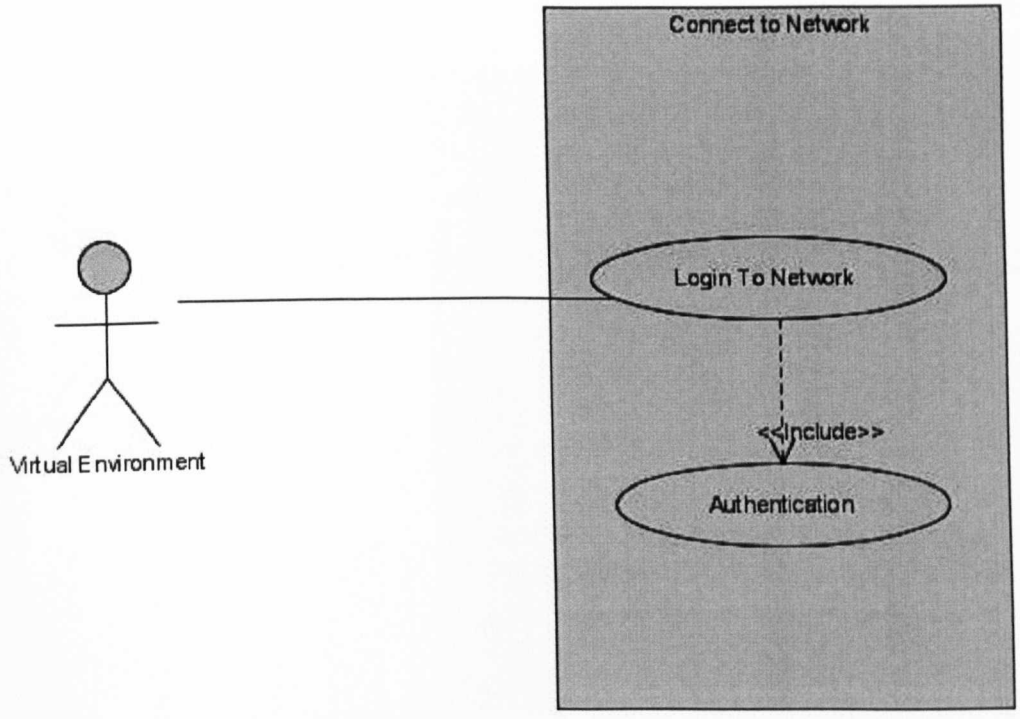


Figure A1.8.2 Connect To Network

Description:

This Use Case illustrates a typical scenario when a virtual environment initially switched on within this framework.

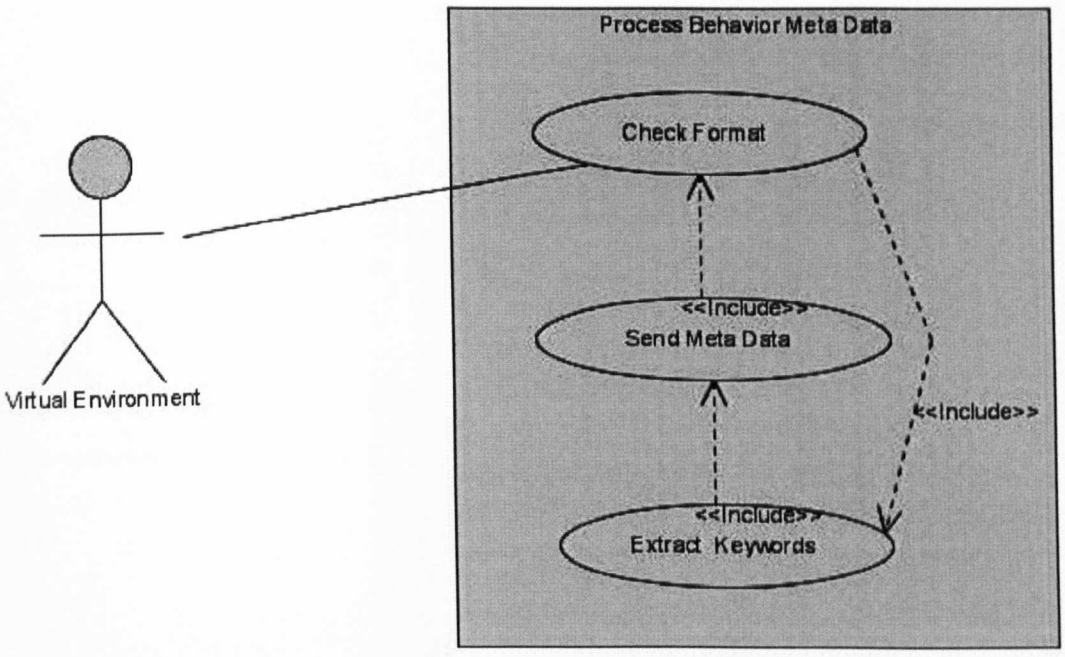


Figure A1.8.3 Process Behaviour Metadata

Description:

This Use Case illustrates a typical scenario when a virtual environment Process behaviour metadata within this framework.

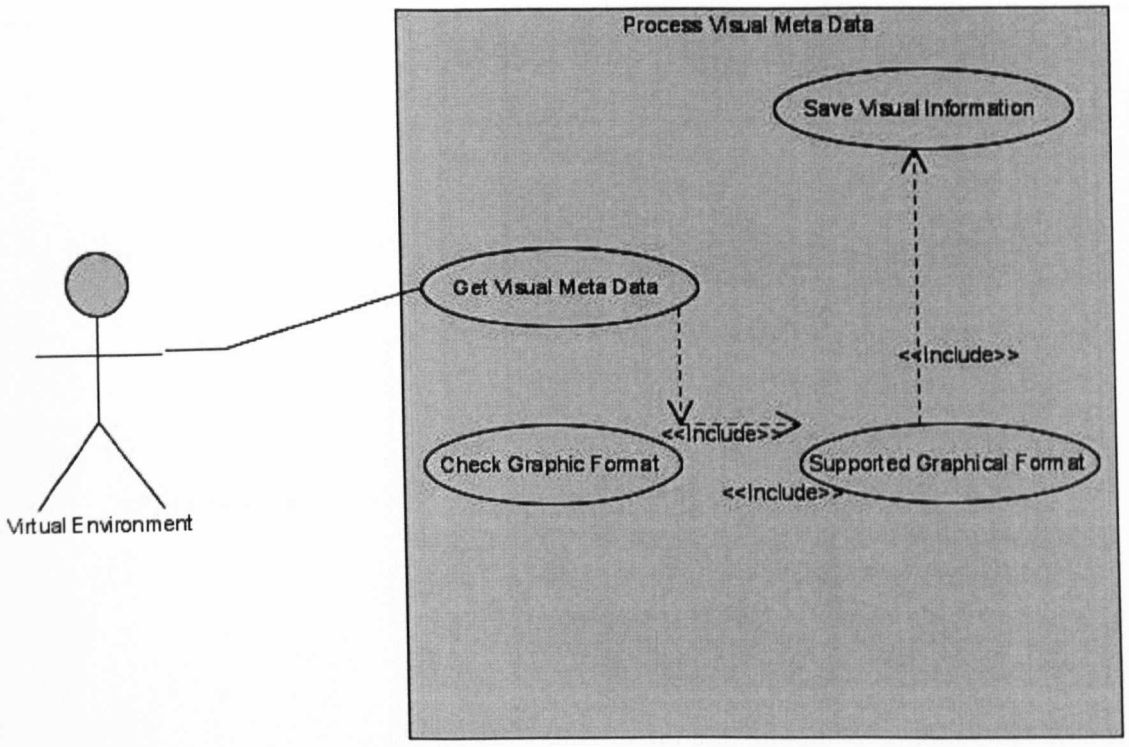


Figure A1.8.4 Process Visual Metadata

Description:

This Use Case illustrates a typical scenario when a virtual environment Process Visual metadata within this framework

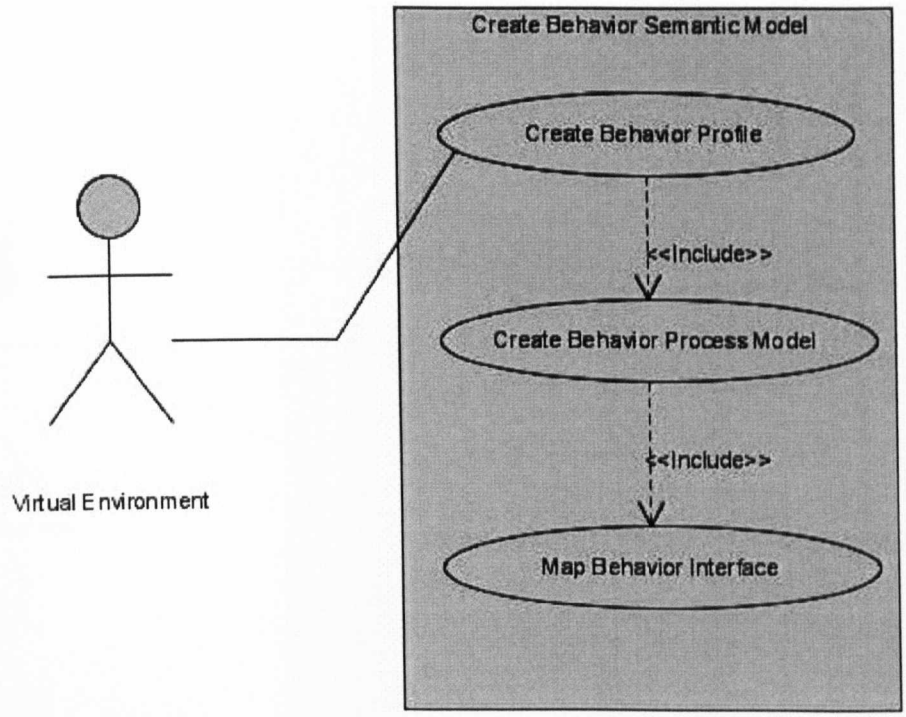


Figure A1.8.5 Create Behaviour Semantic Model

Description:

This Use Case illustrates a typical scenario when a Virtual Environment Create Behaviour Semantic Model within this framework.

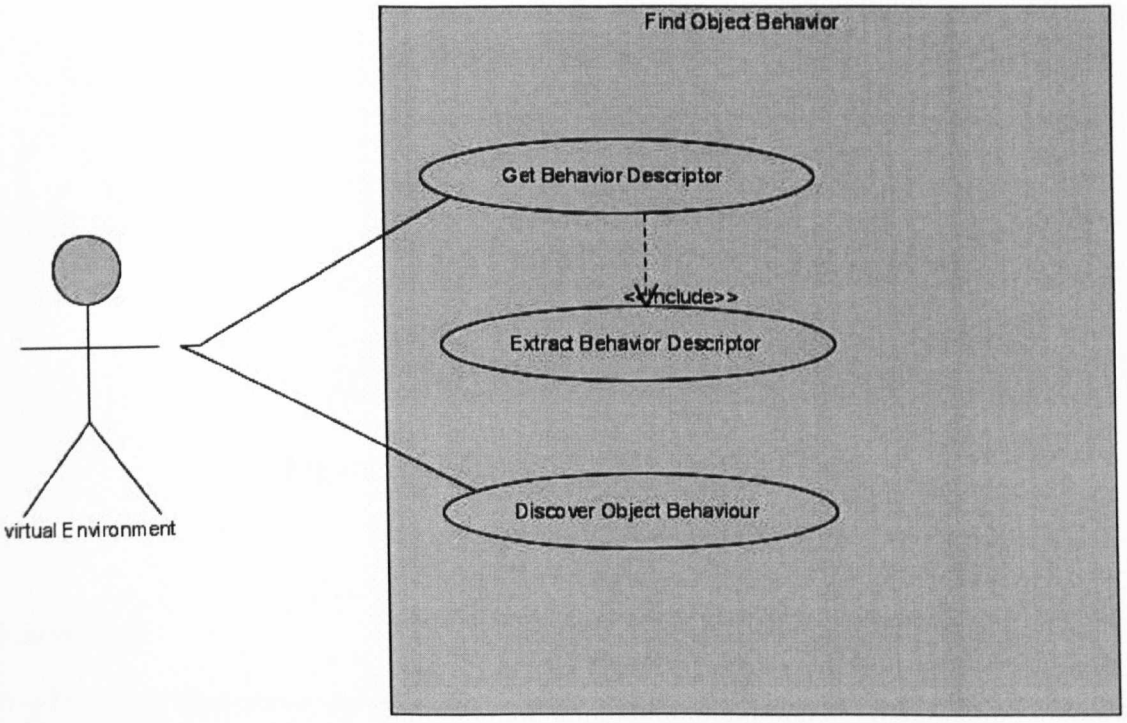


Figure A1.8.6 Find Object Behaviour

Description:

This Use Case illustrates a typical scenario when a virtual environment Discover Behaviour within this framework.

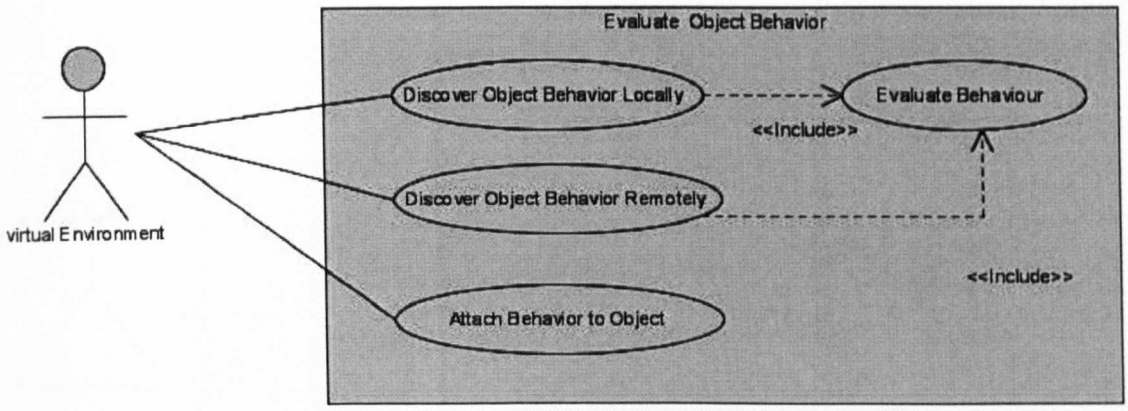


Figure A1.8.7 Evaluate Object Behaviour

Description:

This Use Case illustrates a typical scenario when a virtual environment Discovered Behaviour and it evaluate the behaviour before attaching to an object within this framework.

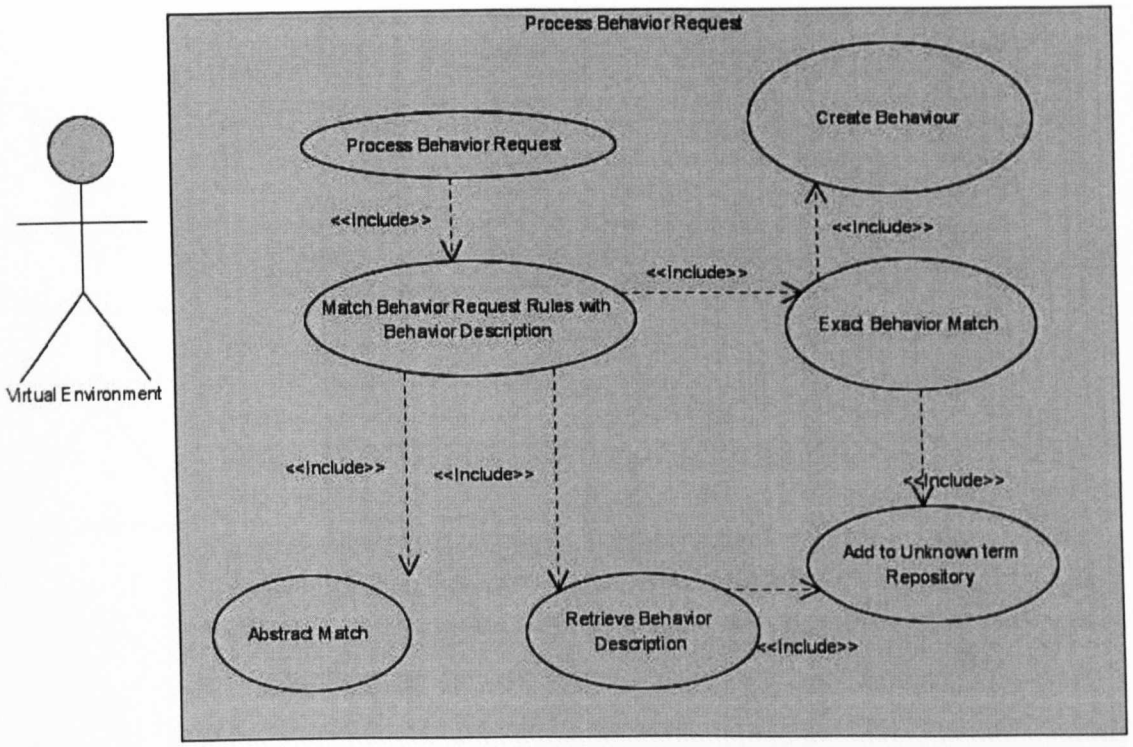


Figure A1.8.8 Process Behaviour Request

Description:

This Use Case illustrates a typical scenario when a virtual environment Process Behaviour Matching process using Semantic description within this framework.

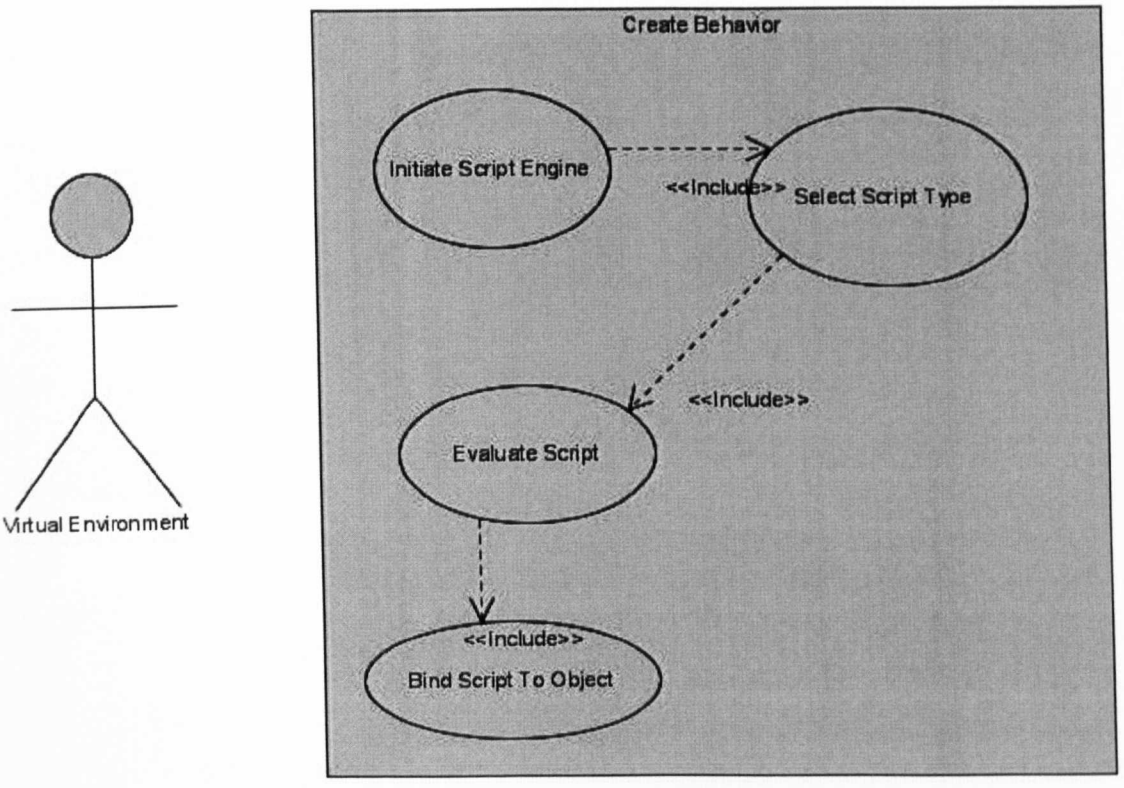


Figure A1.8.9 Scripting Engine

Description:

This Use Case illustrates a typical scenario when a virtual environment Initialize scripting engine to load and evaluate Behaviour within this framework.

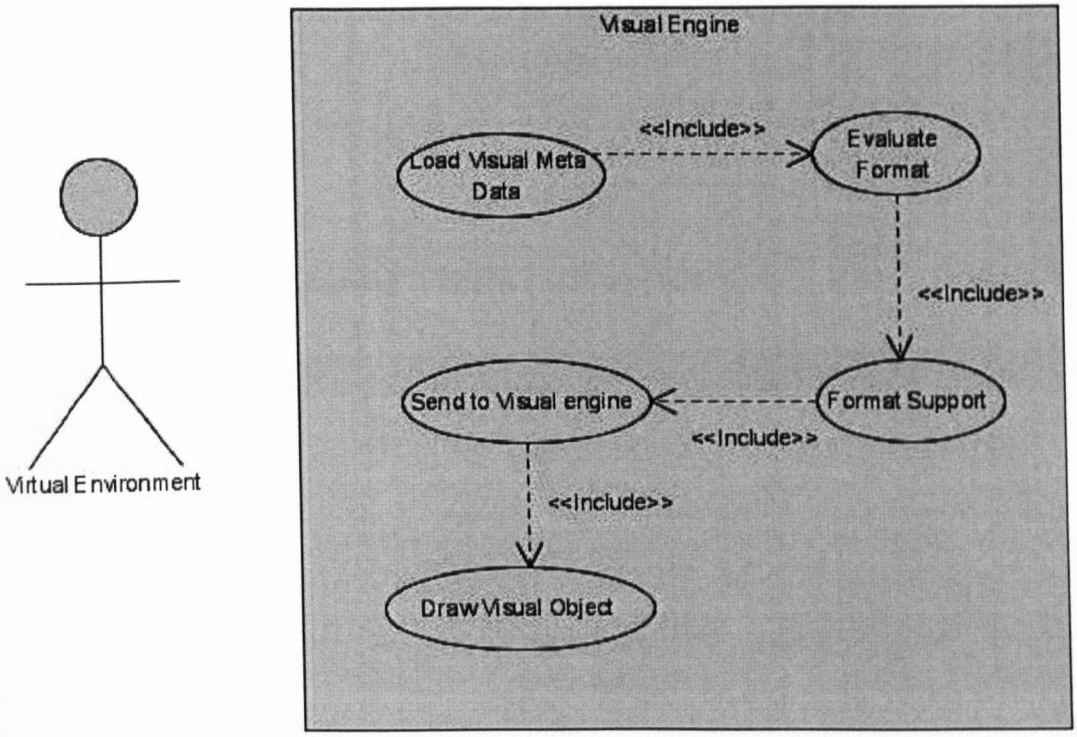


Figure A1.8.10 Visual Engine

Description:

This Use Case illustrates a typical scenario when a virtual environment render object and check its format within this framework.

APPENDIX B: TABLES

Table 8.1 Scenario Parameters

Scenario Parameters	
Network	
Transmitter Range	100 Meter
Bandwidth	100Mbps
Number of Nodes	4
Pack Size	
Software	
OS	Windows XP Professional SP3
Java	jdk1.6.0_07
Fob4JMF	0.4.1
SUF	1.0
Drools	5.0
Scripting Engine	1.6
OWL-S	1.0
Jena	2.0
Protégé-OWL, API	2.1
JME	2.0.1
JMF	2.1.1e
SunSPOT Library	Blue-080827
X-10	1.3
Prototype	
Running Time	1 Hour
Protocols	TCP
Media Transmitted	

APPENDIX C: LOG FILE FOR COMPUTER A

Table 8.2 Log File to add Lamp to Virtual World at run time

1	07-Mar-2011 13:09:14 com.jme.app.BaseGame start
2	INFO: Application started.
3	07-Mar-2011 13:09:15 com.jme.system.PropertiesIO <init>
4	INFO: PropertiesIO created
5	07-Mar-2011 13:09:15 com.jme.system.PropertiesIO load
6	INFO: Read properties
7	07-Mar-2011 13:09:20 com.jme.input.joystick.DummyJoystickInput <init>
8	INFO: Joystick support is disabled
9	07-Mar-2011 13:09:20 com.jme.system.lwjgl.LWJGLDisplaySystem <init>
10	INFO: LWJGL Display System created.
11	07-Mar-2011 13:09:20 com.jme.system.lwjgl.LWJGLDisplaySystem getValidDisplayMode
12	INFO: Selected DisplayMode: 640 x 480 x 16 @60Hz
13	07-Mar-2011 13:09:20 com.jme.system.PropertiesIO save
14	INFO: Saved properties
15	07-Mar-2011 13:09:20 com.jme.app.BaseSimpleGame initSystem
16	INFO: jME version 0.11 beta
17	07-Mar-2011 13:09:22 com.jme.renderer.lwjgl.LWJGLRenderer <init>
18	INFO: LWJGLRenderer created. W: 640H: 480
19	07-Mar-2011 13:09:23 com.jme.app.BaseSimpleGame initSystem
20	INFO: Running on: nv4_disp
21	Driver version: 6.14.10.9131
22	NVIDIA Corporation - GeForce 6200 LE/PCI/SSE2 - 2.0.3
23	07-Mar-2011 13:09:23 com.jme.renderer.AbstractCamera <init>
24	INFO: Camera created.
25	07-Mar-2011 13:09:23 com.jme.util.lwjgl.LWJGLTimer <init>
26	INFO: Timer resolution: 1000 ticks per second
27	07-Mar-2011 13:09:23 com.jme.scene.Node <init>
28	INFO: Node created.
29	07-Mar-2011 13:09:23 com.jme.scene.Node <init>
30	INFO: Node created.
31	07-Mar-2011 13:09:23 com.jme.scene.Node attachChild
32	INFO: Child (FPS label) attached to this node (FPS node)
33	07-Mar-2011 13:09:28 com.jme.scene.Node attachChild
34	INFO: Child (Text Label) attached to this node (FPS node)

```
07-Mar-2011 13:09:28 com.jme.scene.Node attachChild
INFO: Child (Mouse Pointer Indicator) attached to this node (FPS node)
07-Mar-2011 13:09:28 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Leinwand.nocull) attached to this node (Leinwand)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Leinwand) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6.Material.008.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6.Material.009.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6.Material.010.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6.Material.011.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.008.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.009.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.010.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.011.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
```

```
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.008.nocull) attached to this node (PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.009.nocull) attached to this node (PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.010.nocull) attached to this node (PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.011.nocull) attached to this node (PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.008.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.009.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.010.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.011.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.003.nocull) attached to this node (Cube.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.001) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Tuer.Material.012.nocull) attached to this node (Tuer)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Tuer.Material.013.nocull) attached to this node (Tuer)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
```

```
INFO: Child (Tuer) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.002.nocull) attached to this node (Cube)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.nocull) attached to this node (TV-Body)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (TV-Body) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC3.Material.008.nocull) attached to this node (PC3)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC3.Material.009.nocull) attached to this node (PC3)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC3.Material.010.nocull) attached to this node (PC3)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC3.Material.011.nocull) attached to this node (PC3)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC3) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC2.Material.008.nocull) attached to this node (PC2)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC2.Material.009.nocull) attached to this node (PC2)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC2.Material.010.nocull) attached to this node (PC2)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC2.Material.011.nocull) attached to this node (PC2)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC2) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
```

```
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.Material.008.nocull) attached to this node (PC1)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.Material.009.nocull) attached to this node (PC1)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.Material.010.nocull) attached to this node (PC1)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.Material.011.nocull) attached to this node (PC1)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.001.Material.008.nocull) attached to this node
(PC1.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.001.Material.009.nocull) attached to this node
(PC1.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.001.Material.010.nocull) attached to this node
(PC1.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.001.Material.011.nocull) attached to this node
(PC1.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC1.001) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Leinwand.nocull) attached to this node (Leinwand)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Leinwand) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
```

```
INFO: Child (PC6.Material.008.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6.Material.009.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6.Material.010.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6.Material.011.nocull) attached to this node (PC6)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC6) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.008.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.009.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.010.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5.Material.011.nocull) attached to this node (PC5)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC5) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.008.nocull) attached to this node
(PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.009.nocull) attached to this node
(PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.010.nocull) attached to this node
(PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001.Material.011.nocull) attached to this node
(PC4.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.001) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
```

```
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.008.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.009.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.010.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4.Material.011.nocull) attached to this node (PC4)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (PC4) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.003.nocull) attached to this node (Cube.001)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.001) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Tuer.Material.012.nocull) attached to this node (Tuer)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Tuer.Material.013.nocull) attached to this node (Tuer)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Tuer) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.002.nocull) attached to this node (Cube)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (Cube.nocull) attached to this node (TV-Body)
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
INFO: Child (TV-Body) attached to this node (XML loaded scene)
07-Mar-2011 13:09:30 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
```

```
256  INFO: Child (PC3.Material.008.nocull) attached to this node (PC3)
257  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
258  INFO: Child (PC3.Material.009.nocull) attached to this node (PC3)
259  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
260  INFO: Child (PC3.Material.010.nocull) attached to this node (PC3)
261  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
262  INFO: Child (PC3.Material.011.nocull) attached to this node (PC3)
263  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
264  INFO: Child (PC3) attached to this node (XML loaded scene)
265  07-Mar-2011 13:09:30 com.jme.scene.Node <init>
266  INFO: Node created.
267  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
268  INFO: Child (PC2.Material.008.nocull) attached to this node (PC2)
269  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
270  INFO: Child (PC2.Material.009.nocull) attached to this node (PC2)
271  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
272  INFO: Child (PC2.Material.010.nocull) attached to this node (PC2)
273  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
274  INFO: Child (PC2.Material.011.nocull) attached to this node (PC2)
275  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
276  INFO: Child (PC2) attached to this node (XML loaded scene)
277  07-Mar-2011 13:09:30 com.jme.scene.Node <init>
278  INFO: Node created.
279  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
280  INFO: Child (PC1.Material.008.nocull) attached to this node (PC1)
281  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
282  INFO: Child (PC1.Material.009.nocull) attached to this node (PC1)
283  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
284  INFO: Child (PC1.Material.010.nocull) attached to this node (PC1)
285  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
286  INFO: Child (PC1.Material.011.nocull) attached to this node (PC1)
287  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
288  INFO: Child (PC1) attached to this node (XML loaded scene)
289  07-Mar-2011 13:09:30 com.jme.scene.Node <init>
290  INFO: Node created.
291  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
292  INFO: Child (PC1.001.Material.008.nocull) attached to this node
     (PC1.001)
293  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
```

```
294  INFO: Child (PC1.001.Material.009.nocull) attached to this node (PC1.001)
295  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
296  INFO: Child (PC1.001.Material.010.nocull) attached to this node (PC1.001)
297  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
298  INFO: Child (PC1.001.Material.011.nocull) attached to this node (PC1.001)
299  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
300  INFO: Child (PC1.001) attached to this node (XML loaded scene)
301  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
302  INFO: Child (XML loaded scene) attached to this node (XML loaded scene)
303  07-Mar-2011 13:09:30 com.jme.scene.Node attachChild
304  INFO: Child (XML loaded scene) attached to this node (rootNode)
305  Connected with data source to load objects2011-03-07 13:09:30
306  07-Mar-2011 13:09:30 uk.ac.ljmu.nal.DataLayer.DataAccess.gameObjectData FillList
307  INFO: Object loaded from database
308  07-Mar-2011 13:09:30 uk.ac.ljmu.nal.DataLayer.DataAccess.gameObjectData FillList
309  INFO: Loading object SunSpot
310  07-Mar-2011 13:09:30 uk.ac.ljmu.nal.DataLayer.DataAccess.gameObjectData FillList
311  INFO: Loading object dvd
312  07-Mar-2011 13:09:34 uk.ac.ljmu.nal.DataLayer.BusinessLogic.gameObjectManager getGameObjects
313  INFO: Firing rules for SunSpot
314  07-Mar-2011 13:09:34 uk.ac.ljmu.nal.DataLayer.BusinessLogic.gameObjectManager getGameObjects
315  INFO: Firing rules for dvd
316  07-Mar-2011 13:09:34 com.jme.scene.Node <init>
317  INFO: Node created.
318  07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
319  INFO: Child (Box) attached to this node (SunSpot)
320  07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
321  INFO: Child (SunSpot) attached to this node (rootNode)
```

```
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Playbutton.Material.031.nocull) attached to this node (Play)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Playbutton.Material.022.nocull) attached to this node (Play)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Play) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Stopsign.Material.028.nocull) attached to this node (Stop)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Stopsign.Material.027.nocull) attached to this node (Stop)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Stop) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Cone) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Forward.Material.006.nocull) attached to this node (Forward)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Forward.Material.026.nocull) attached to this node (Forward)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Forward) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
```

```
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Rewind.Material.021.nocull) attached to this node (Rewind)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Rewind.Material.025.nocull) attached to this node (Rewind)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Rewind) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Pause.Material.029.nocull) attached to this node (Pause)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Pause.Material.030.nocull) attached to this node (Pause)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Pause) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Cube.004) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Mediaplayer.nocull) attached to this node (Mediaplayer)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Mediaplayer) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Playbutton.Material.031.nocull) attached to this node (Play)
```

```
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Playbutton.Material.022.nocull) attached to this node (Play)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Play) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Stopsign.Material.028.nocull) attached to this node (Stop)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Stopsign.Material.027.nocull) attached to this node (Stop)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Stop) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Cone) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Forward.Material.006.nocull) attached to this node (Forward)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Forward.Material.026.nocull) attached to this node (Forward)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Forward) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Rewind.Material.021.nocull) attached to this node (Rewind)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Rewind.Material.025.nocull) attached to this node (Rewind)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Rewind) attached to this node (XML loaded scene)
```

```
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Pause.Material.029.nocull) attached to this node (Pause)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Pause.Material.030.nocull) attached to this node (Pause)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Pause) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Cube.004) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node <init>
INFO: Node created.
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Mediaplayer.nocull) attached to this node (Mediaplayer)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (Mediaplayer) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (XML loaded scene) attached to this node (XML loaded scene)
07-Mar-2011 13:09:34 com.jme.scene.Node attachChild
INFO: Child (XML loaded scene) attached to this node (rootNode)
07-Mar-2011 13:09:34 uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
INFO: new object lamp has been detected
07-Mar-2011 13:09:35 uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
INFO: loading description information from lamp
07-Mar-2011 13:09:39 uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
INFO: loading completed
07-Mar-2011 13:09:39 uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
INFO: Object behaviour not found for lamp
07-Mar-2011 13:09:39
```

	uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
451	INFO: searching on the network
	07-Mar-2011 13:09:44
452	uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
453	INFO: found behaviour for lamp read to transfer
	07-Mar-2011 13:09:44
454	uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
455	INFO: Behaviour Transfer started
	07-Mar-2011 13:09:49
456	uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
457	INFO: Behaviour Transfer Completed
	07-Mar-2011 13:09:49
458	uk.ac.ljmu.nal.visualEngine.visualizationEngine validateBehaviour
459	INFO: validating behaviour
	07-Mar-2011 13:09:50
460	uk.ac.ljmu.nal.visualEngine.visualizationEngine validateBehaviour
461	INFO: Behaviour has no error
	07-Mar-2011 13:09:50
	uk.ac.ljmu.nal.visualEngine.visualizationEngine
462	attacheBehaviourToObject
463	INFO: Behaviour has succesfully attached to object lamp
	07-Mar-2011 13:09:50
464	uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
465	INFO: Rendering started for lamp
	07-Mar-2011 13:09:55
466	uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
467	INFO: Rendering is completed for lamp

Table 8.3 Log file To add lamp to virtual world at run time

	07-Mar-2011 16:38:36
	uk.ac.ljmu.nal.visualEngine.visualizationEngine
1	ListeningToNewObjects
2	INFO: new object lamp has been detected
	07-Mar-2011 16:38:37
	uk.ac.ljmu.nal.visualEngine.visualizationEngine
3	ListeningToNewObjects

4	INFO: loading description information from lamp
5	07-Mar-2011 16:38:41 uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
6	INFO: loading completed
7	07-Mar-2011 16:38:41 uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
8	INFO: Object behaviour not found for lamp
9	07-Mar-2011 16:38:41 uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
10	INFO: searching on the network
11	07-Mar-2011 16:38:45 uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
12	INFO: found behaviour for lamp read to transfer
13	07-Mar-2011 16:38:45 uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
14	INFO: Behaviour Transfer started
15	07-Mar-2011 16:38:50 uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
16	INFO: Behaviour Transfer Completed
17	07-Mar-2011 16:38:50 uk.ac.ljmu.nal.visualEngine.visualizationEngine validateBehaviour
18	INFO: validating behaviour
19	07-Mar-2011 16:38:51 uk.ac.ljmu.nal.visualEngine.visualizationEngine validateBehaviour
20	INFO: Behaviour has no error
21	07-Mar-2011 16:38:51 uk.ac.ljmu.nal.visualEngine.visualizationEngine attacheBehaviourToObject
22	INFO: Behaviour has succesfully attached to object lamp
23	07-Mar-2011 16:38:51 uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
24	INFO: Rendering started for lamp

	07-Mar-2011 16:38:56
25	uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
26	INFO: Rendering is completed for lamp

Table 8.4 Log file To add lamp to virtual world at run time

	07-Mar-2011 16:45:44
1	uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
2	INFO: new object lamp has been detected
	07-Mar-2011 16:45:45
3	uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
4	INFO: loading description information from lamp
	07-Mar-2011 16:45:49
5	uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
6	INFO: loading completed
	07-Mar-2011 16:45:49
7	uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
8	INFO: Object behaviour not found for lamp
	07-Mar-2011 16:45:49
9	uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
10	INFO: searching on the network
	07-Mar-2011 16:45:56
11	uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
12	INFO: found behaviour for lamp read to transfer
	07-Mar-2011 16:45:56
13	uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
14	INFO: Behaviour Transfer started
	07-Mar-2011 16:46:01
15	uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
16	INFO: Behaviour Transfer Completed
	07-Mar-2011 16:46:01
17	uk.ac.ljmu.nal.visualEngine.visualizationEngine

	validateBehaviour
18	INFO: validating behaviour
19	07-Mar-2011 16:46:01 uk.ac.ljmu.nal.visualEngine.visualizationEngine validateBehaviour
20	INFO: Behaviour has no error
21	07-Mar-2011 16:46:02 uk.ac.ljmu.nal.visualEngine.visualizationEngine attacheBehaviourToObject
22	INFO: Behaviour has succesfully attached to object lamp
23	07-Mar-2011 16:46:02 uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
24	INFO: Rendering started for lamp
25	07-Mar-2011 16:46:07 uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
26	INFO: Rendering is completed for lamp

Table 8.5 sunSpot Object Log file

1	09-Mar-2011 14:53:06 uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
2	INFO: new object sunSpot has been detected
3	09-Mar-2011 14:53:07 uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
4	INFO: loading description information from sunSpot
5	09-Mar-2011 14:56:07 uk.ac.ljmu.nal.visualEngine.visualizationEngine ListeningToNewObjects
6	INFO: loading completed
7	09-Mar-2011 14:56:07 uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
8	INFO: Object behaviour not found for sunSpot
9	09-Mar-2011 14:56:07 uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
10	INFO: searching on the network
11	09-Mar-2011 14:56:14 uk.ac.ljmu.nal.visualEngine.visualizationEngine SearchObjects
12	INFO: found behaviour for sunSpot read to transfer
13	09-Mar-2011 14:56:14 uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
14	INFO: Behaviour Transfer started
15	09-Mar-2011 14:56:21 uk.ac.ljmu.nal.visualEngine.visualizationEngine transferBehaviour
16	INFO: Behaviour Transfer Completed

17	09-Mar-2011 14:56:21 uk.ac.ljmu.nal.visualEngine.visualizationEngine validateBehaviour
18	INFO: validating behaviour
19	09-Mar-2011 14:56:22 uk.ac.ljmu.nal.visualEngine.visualizationEngine validateBehaviour
20	INFO: Behaviour has no error
21	09-Mar-2011 14:56:22 uk.ac.ljmu.nal.visualEngine.visualizationEngine attacheBehaviourToObject
22	INFO: Behaviour has succesfully attached to object sunSpot
23	09-Mar-2011 14:56:22 uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
24	INFO: Rendering started for sunSpot
25	09-Mar-2011 14:56:24 uk.ac.ljmu.nal.visualEngine.visualizationEngine renderObject
26	INFO: Rendering is completed for sunSpot