A FRAMEWORK FOR SELF-ADAPTIVE NETWORKED APPLIANCES

By

Paul Fergus MSc., BSc., MBCS

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

December 2005

This thesis is dedicated to my father,
Geoffrey Earnest Fergus 1949 - 1997

ACKNOWLEDGEMENTS

There have been many contributing factors associated with the completion of my PhD, the most important being all the people who have helped me. I would like to thank Professor Madjid Merabti for giving me the opportunity to fulfil my dreams; for believing in me; providing me with invaluable knowledge and above all for being a good friend through the good and bad times – I am forever indebted to you Professor Merabti. I would like to thank Dr Martin Hanneghan for his continued support and guidance throughout my PhD. I will remember the numerous conversations we have had over the past three years with great fondness. I would like to thank him for sharing his in-depth technical and academic knowledge for which this thesis would not have been possible. I would also like to thank him for being a good friend.

I would like to thank my dad who, before he died, urged me to carry on with my studies when I wanted to quit. I would also like to thank a very special family who have recently come into my life and supported me during the final stages of my PhD; Lorna, Sasha-Lei, Lillian and Brian Bracegirdle. I would also like to thank the cats Kami, Smokey, Minnie, Tigger, and Charlie, for making me laugh when I needed it.

Last, but by no means least, I would like to thank all my friends who supported me through my studies, which include Nicholas and Annette Hodder, Neil and Dawn Beaumont, Sue and Ken Richardson, Hilton and Rona McCabe, Shaun and Christopher Bennett, Ray Brizell and Jenny Leavitt. I would like to extend a special thank you to Omar Abuelmatt'ti, Anirach Mingkhwan, Gurleen Arora, Henry Chang, David Llewellyn-Jones, John Haggerty, Bob Askwith, Arshad Mahammud, Huma Javed, Azzelarabe Taleb-bendiab, Qi Shi, Janette Skentelbery, and Carol Oliver to name a few, for all their help and support over the years.

ABSTRACT

The proliferation of home appliances and the complex functions they provide make it ever harder for a specialist, let alone an ordinary home user, to configure and use them. Imagine your home environment, more specifically your living room, and the devices it contains. It is more than likely that it has a DVD player, Widescreen or Plasma TV, a surround sound speaker system, and a HiFi. Now imagine the time you bought your DVD player and tried to integrate it with your existing home appliance configuration. After taking the DVD player out of the box you will have connected all the wires and tuned in your TV. This whole process may have taken several hours and it is likely the configuration was not correct first time. These kinds of experiences are becoming increasingly more common because devices and their associated configurations are becoming more complex.

Now image a future environment whereby you take the DVD player out of the box, switch it on, and it just works. You put your DVD movie into the player, press play and the video is displayed on your TV, whilst the sound is directed to the surround sound speaker system. You do not have to manually connect the player to any external devices and you do not have to tune in your TV. When the DVD player is switched on it automatically communicates with all other devices needed within the home via its wireless network interface. When the play button is pressed all the devices are combined to form a home entertainment system and released when the player no longer needs them.

In trying to achieve this, many challenges need to be addressed, which include service-oriented networking; service discovery; device capability matching; dynamic service composition; and device self-adaptation. Overcoming these challenges will allow mechanisms to be developed that simplify the configuration and management tasks associated with next generation networked appliances.

In this thesis we address these challenges using a new framework we have developed called the Networked Appliance Service Utilisation Framework. Our framework allows heterogeneous devices to be seamlessly interconnected and operated with little human intervention. The operational functions provided by different appliances are dispersed within the network and used to create high-level applications. Devices are interconnected using a service-oriented middleware and discovered and combined using machine-processable descriptions. Our framework takes into account the capabilities devices support and provides self-adaptation mechanisms to manage device configurations automatically. We have successfully developed a working prototype that implements an Intelligent Home Environment, which is used to quantitatively evaluate our framework.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND TERMS

API	Application Program Interface
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
DistrES	Distributed Emergent Semantics – evolves semantic structures
DiSUS	Distributed Semantic Unstructured Services – P2P implementation
DL	Description Logics
EPG	Electronic Program Guides
GENA	General Event Notification Architecture
HES	Home Electronic System
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
NASUF	Networked Appliance 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

In recent years, with the growth of personal computer usage and the Internet, networked computers have become more widely used in more diverse applications. As this trend continues, we can expect ordinary everyday appliances to become part of these networks, and networked devices will become pervasive and often invisible to the users.

As connectivity at broadband speeds becomes an integral part of our household infrastructure, it is envisaged that every device will have a network interface that allows it to be accessed and controlled from anywhere in the world. This idea is generating a great deal of interest and a number of research initiatives have been proposed that include on-demand multimedia services [France Telecom 2005], home automation through wireless sensor networks and remote control of home appliances through immersive technologies and global communications [Koumpis 2005]. Sound business models are being developed to realise such applications based on market and user needs that will map the future direction of Internet and home technologies.

We are already seeing this transition in home entertainment systems, allowing for a greater level of sophistication in how users interact with multimedia service subscriptions and the devices they have installed. The provision to monitor and control the home using TV sets and set-top boxes has advanced rapidly in recent years because the TV is considered as the central appliance within a typical home environment [Evans 2001, Marshall 2001, Bhatti 2002]. Interactive-TV and real-time communication during live broadcasts using advances in global communications and mobile devices have become common place. The ability to pause live TV and personalise multimedia services has given users greater control over how and when they interact with digital entertainment. Furthermore we are seeing a convergence between personal computing and home entertainment systems. The advent of media centre set-top boxes allow users to connect the devices they own and access a plethora of on-line multimedia services, via their broadband connection, such as digital radio, electronic programme guides (EPG), on-demand Internet TV, on-line gaming, including services associated with modern day computing such as email and instant messaging. However, this

said, we are at a crossroads whereby configuring and managing next generation networked appliances and home networks will become increasingly more complex. As we will argue in this thesis existing approaches lack scalability and sound business models to fully utilise new technological shifts and as such alternative mechanisms are required.

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 and all concepts relating to its construction are clearly defined, which includes networked appliances, home networking, service-oriented architectures, service discovery, dynamic service composition and self-adaptation. 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.

1.2 Networked Appliances and Home Networking

For more than a decade, home and building automation and networking have received much consideration by homeowners, industry and academic researchers [Dutta-Roy 1999, Siuru 2000]. This includes the introduction of a wide spectrum of wired and wireless infrastructures and network protocols such as LonWorks, CEBus, SmartHouse, VHN, HomePNA, HomePnP, IEEE1394 (Firewire), X-10, IrDA, IEEE802.11b, Bluetooth and HyperLAN/2 [Rose 2001]. However despite the long list of advantages they provide, several challenges that need to be considered, most notably, interoperability [Abuelma'atti 2002a, Zahariadis 2002] and the difficulties associated with the integration of combined functionalities. In Figure 1.1 a typical home environment is illustrated; the challenge is to combine devices from different domains, *i.e.* broadcast, internet and mobile, and disperse their operational functions within the network so that they can be used by any device within those domains.

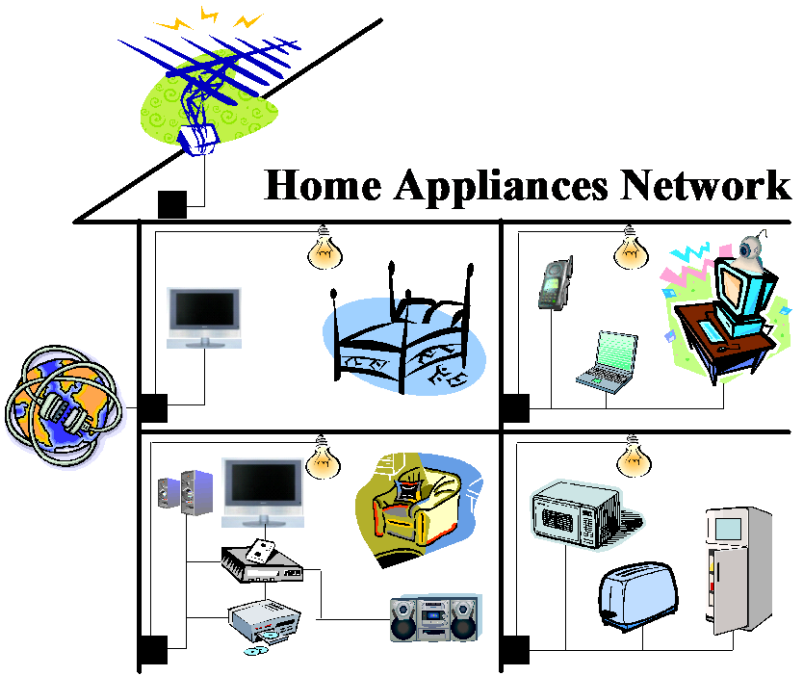

Figure 1.1 Networking home appliances

Many industry efforts have evolved to create interworking solutions, which include the Home Electronic System (HES) [Pattenden 2001], Home Audio-Video Interoperability (HAVi) [HAVI 2003], Universal Plug and Play (UPnP) [Miller 2001, Microsoft Corp. 2005] and it's Intel Digital Home implementation [Intel 2003] and the Open Services Gateway Initiative (OSGi) [Marples 2001]. Additionally, research efforts within networked appliances and service discovery disciplines are trying to provide solutions, which define scenarios for new and emerging network configurations [Cheng 2000, Minoh 2001]. For example, the provision of home monitoring and control systems from within TV sets and set-top boxes has advanced rapidly in recent years because the TV is considered the central appliance within a typical home environment [Evans 2001, Marshall 2001, Bhatti 2002].

The main goal is to ensure user acceptance and provide flexible systems that will become integrated within the household infrastructure. This transition mirrors the evolutionary process undertaken within personal computing and wide area communications, whereby it is now difficult to imagine using a computer without Internet access. Given the success of this transition, home networking platforms aim to achieve the same level of acceptance whereby it will be impossible to imagine home appliances without Internet access.

Many research initiatives are trying to move away from bespoke solutions by combining embedded systems with the Internet allowing more complex solutions to be developed. The complexity itself is a by-product of heterogeneity and the dynamic nature associated with networks that resist any form of control. However putting complexity aside there is still a need to promote this integration because bespoke development is too expensive and too

limiting for innovative applications. This is clearly a trade off between inflexible, but reliable, and flexible but unreliable systems. The end goal must be flexibility based on sound engineering principles that produce self-adaptive middleware frameworks that enable heterogeneous networks, devices and services to be seamlessly interconnected.

Although there are many solutions that allow devices to be interconnected within the home environment, diminutive advances have been made to abstract the complexity away from this process. Technology is evermore pervasive and effectively managing it is not an easy task. Advances made in global communications and service-oriented architectures promise to provide a platform that realises a seamless integration between heterogeneous devices, however few solutions have produced any convincing results. The challenge is get different appliances built to different specifications, to work together.

1.3 Structured and Unstructured Services

Visualise a high street shopping area, which is a simple outdoor environment. The street is full of shops, restaurants, street vendors and other people. We pop in and out from one shop to another, buy a quick snack from a street vendor – here today gone tomorrow – and greet people we know. All of these activities happen within our focal view. Devices within real-world environments have to work the same way as this shopping area analogy. This provides devices, with the ability to interact and use services in the same way people interact with shops within real-world environments.

What emerges from this analogy for service usage is defined as an Information Space [Mingkhwan 2002] and illustrated in Figure 1.2. Information Space is the concept of integrating information and services from the environment a device has access to. By considering the device as the centre of surrounding information and services we find that, in reality, the environment that the device moves into provides services. The ability to select and use these services to offer the maximum flexibility for the device is of paramount importance.

The need for an integrated information space requires the unification of wired and wireless networks and their services. In particular, the challenge is to bring together services within ad hoc networks such as Bluetooth and infrastructure networks like the Internet [Mingkhwan 2003]. Devices provide services throughout the Information Space using middleware that interconnects infrastructure networks and ad hoc networks together.

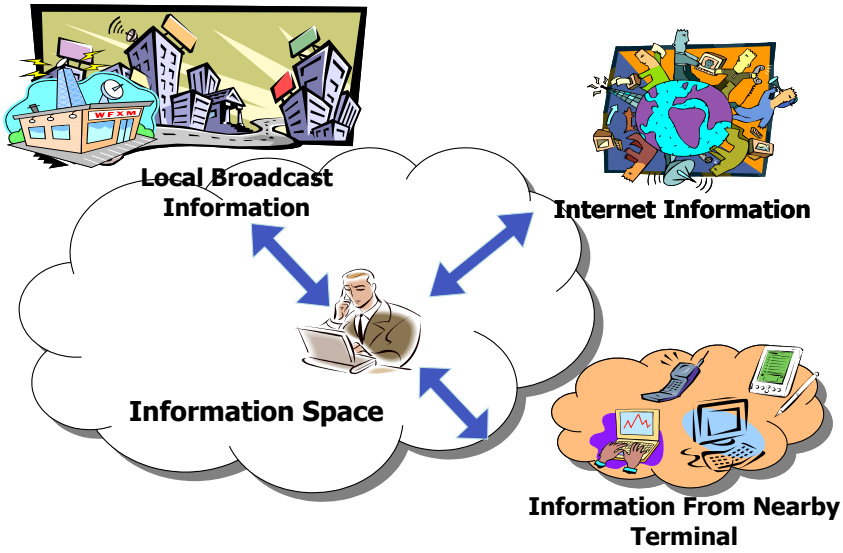

Figure 1.2 Information Space

Services within an Information Space can be described as *structured* and *unstructured* and are defined as follows:

- *Structured Services* use third party software to register and advertise functions the peer provides, e.g., Directory, Proxy and Naming Services. These kinds of services typically have complex structures, such as network connectivity, database access and multimedia functions.
- *Unstructured Services* provide services independent of any kind of third party intervention. This concept is based on a simple service definition, such as a kiosk that provides quick information, a TV remote control that simply changes the channel or a file-sharing application that exchanges digital content.

There are an increasing number of structured services available to users over the Internet and ad hoc networks, yet unstructured services remain far behind. Internet-based structured services like JINI [JINI Technology 2005] and UDDI [Paolucci 2002b, WebMethods 2003] are already well defined; however they are incapable of providing services within dynamically changing network environments. This limitation can be simplified by situating services within the Information Space, using decentralised networking concepts [Parameswaran 2001].

The challenge is to distribute services within the network and discover them without having to rely on third party registries. This requires mechanisms to dynamically discover and utilise what services are available within the devices immediate and extended environment. This is important if we are to ensure flexibility and provide mechanisms for true zero-configuration.

1.4 Improving Service Discovery

Although services will become an important enabling technology several other difficulties need to be overcome. The problem is that current service-oriented solutions ignore the fact that the service space will become increasingly large. As such existing approaches fail to discover services based on what a service is capable of doing. Consequently selecting the correct service to satisfy our needs will become increasingly more important and lessons need to be learnt from the problems experienced within the Web in terms of accurately finding content. As such the challenge is to describe services better so that devices can reason over what they require and what services are available.

Although several standards exist to describe and discover services, they fail to address interoperability between open standards and the vocabularies used. Their efforts strive to develop universally agreed vocabularies that describe services homogeneously however this is a very difficult challenge, if not impossible. Researchers within the Semantic Web community are trying to address this limitation by developing an alternative approach that enables semantic interoperability between different vocabularies using machine-processable semantics. However the major difficulties that still need to be addressed are how semantic structures are created, distributed, managed, and evolved over time.

As such, environments need to support mechanisms that enable knowledge to emerge whereby each device is treated as a self-governing knowledge node that is free to share and discover ontological structures. The challenge is to enable a distributed environment that provides the following functions:

- A mechanism that enables the representation and discovery of semantic information.
- A mechanism that captures the general consensus within responses received from devices in terms of ontological structures.
- Algorithms that evolve and merge semantic knowledge over time.

Several research initiatives are trying to create techniques for “intelligent” information gathering [Heflin 2000, Stephens 2001, Fensel 2002, Siebes 2002, Stephens 2003] to allow devices to share knowledge in a distributed network analogous to the way people learn and acquire new knowledge through communication. However mechanisms still need to be developed that codify this human activity and provide knowledge management solutions that distance themselves from ontology construction mechanisms based on the opinions of small centralised ontology consortiums. Devices need to evolve their internal knowledge structures to conceptually understand the vocabularies used within the network in order to better discover services that are semantically described. This will allow rich ontological structures to

emerge over time as fragmented knowledge structures are discovered and merged by devices within the network.

The challenge is to semantically discover and evolve ontological structures within distributed environments based on localised ontology structures and general consensus. The key technique needs to focus on merging information based on general consensus, found within all responses received from the network, for a particular query. As such techniques to determine the general consensus need to be devised, *i.e.* techniques based on evolutionary programming [Langton 1996], or statistics.

1.5 Composing Networked Appliances Automatically

It is apparent that connecting networked appliances is becoming increasingly more difficult because their associated configuration is more complex. The challenge is to automate the process and enable devices to perform any required configuration or management themselves. Many research initiatives are trying to address this using a number of different approaches, which include manual, semi and automated device and service composition techniques [McIlraith 2001, Narayanan 2002, Chakraborty 2003, Chen 2003, Medjahed 2003, Sirin 2003, Sycara 2003, Fujii 2004, Madhusudan 2004, Milanovic 2004]. These solutions are human-centric where services, designed to abstract device functions as network components, are composed via user defined interfaces.

These solutions lack scalability and it is quickly becoming apparent that alternative mechanisms are required that allow networked appliances to be dynamically composed based on user requirements. The goal is to create value-added operational functionality that, when combined, produce functions that could not be performed by one device alone. These research initiatives are firmly embedded within the Networked Appliance and Semantic Web Service community where services can be discovered, composed and executed using service ontologies. Although these research initiatives have produced some interesting results, there is no one solution that truly allows devices to be dynamically composed devoid of any human intervention. Users can discover and integrate services using workflows languages such as BPEL4WS and WSFL, however mechanisms that allow services to be dynamically discovered and composed in an ad hoc fashion, are far from a reality.

Alternative mechanisms need to be developed that overcome the inherent restrictive nature of workflow standards that allow service descriptions to semantically describe what devices require and what they provide. The challenge is to combine service technologies with machine-processable semantics to automatically interconnect devices using high-level semantics that loosely bind devices together. This will enable true zero-configuration,

whereby devices automatically integrate themselves within the environment and link together using conceptual information about what the device does and what it needs.

1.6 Flexible Networked Appliances and Self-Adaptation

Currently, connecting and managing device configurations, is inherently a manual process, and as highlighted in this chapter it is becoming increasingly more difficult for IT specialists and home users alike. It is no longer acceptable to just accept this problem because we are reaching a point whereby the effort required will surpass the need to buy networked appliances and implement home networking solutions.

Self-adaptive mechanisms need to be developed that allow devices to automatically form relationships with each other with little or no human interaction. For example, in the future when you buy a DVD player and take it out of the box, it will automatically integrate itself with existing device configurations, once it has been switch it on. When you put a movie into the player and press play it automatically displays on your TV and outputs sound via your surround sound speaker system. Extending this idea further the player may only process MPEG-2 media formats. If you try to watch a movie that uses an Xvid encoding, (a format your machine does not support) the player will try to resolve this conflict by automatically discovering and downloading the appropriate codec or using an intermediary service to transcode the data into MPEG-2, via its Internet connection. This will allow devices to extend their functionality beyond what they where initially designed to do by forming relationships with other devices and services within the network.

Such a vision provides considerable benefits to the consumer by allowing networked appliances to be automatically integrated and evolved. However, currently devices and middleware solutions do not provide any mechanism to achieve this. The challenge is to develop new mechanisms capable of automatically integrating devices and managing any conflicts within device configurations that may occur. The underlying implementation details need to be abstracted, thus enabling all devices and services to appear homogeneous within and across different domains.

1.7 Scope of the research

The aim of this thesis is to develop a new framework, as illustrated in Figure 1.3 that allows the operational functions provided by different appliances to be dispersed within the network and used to create high-level applications. The framework will use a service-oriented middleware to discover and combine devices using machine-processable descriptions that allow devices and functions to be selected based on application requirements. This framework

will take into account the capabilities devices support and provide self-adaptation mechanisms to manage device configurations automatically.

Although security and transport protocol interoperability are important requirements they are not seen as pertinent to proving the ideas presented in this thesis. The framework is a flexible platform that can allow any additional requirements to be plugged in as and when they are needed. As such the Networked Appliance Service Utilisation Framework, as illustrated in Figure 1.3 is only considered within the remainder of this thesis.

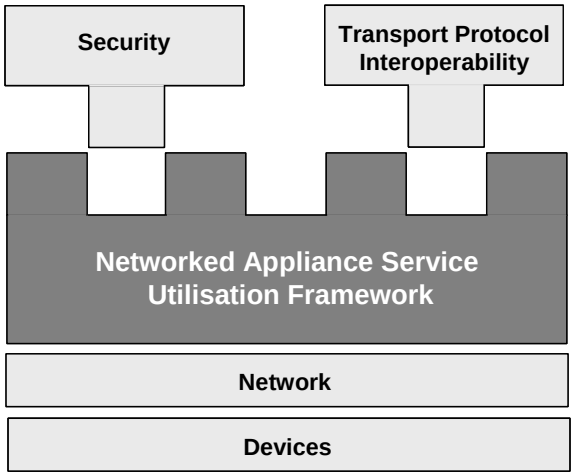

Figure 1.3 Proposed Framework

Using this framework several key requirements are addressed within this thesis, which encompass advances made in the areas of service-oriented networking, networked appliances, service discovery, dynamic service composition and self-adaptation. It does not consider the aforementioned disciplines in isolation but rather investigates how they can be combined and extended to create a new type of framework capable of seamlessly interconnecting devices.

1.8 Project Requirements

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

- The functions offered by complex devices need to be published as independent services so that they can be discovered and utilised by other devices within the network.
- Devices must have the ability to offer zero or more framework services. If a service is not hosted by the device then it must be capable of discovering and using the service remotely within the network. Framework services must be discovered and bound to before the device is rendered fully functional.
- It is fundamental that services offered by devices are discovered without forcing the device or the services it provides to register with centralised authorities. Once devices

are switched on they must be capable of offering their services without being constrained by a third-party service registry.

- Service *descriptions* and service *requests* must be based on machine-processable semantics to successfully determine what services are relevant and what are not. This brings with it additional challenges. The vocabularies used by different device manufacturers will be different and the structure of the concepts themselves will vary. Therefore mechanisms need to be developed that allow devices to dynamically create a semantic interoperability bridge between terms that are syntactically distinct but semantically equivalent. This mechanism must allow devices to discover other devices and services within their environment and dynamically learn the different terminologies they use. During the learning process vocabularies must be evolved based on general consensus, whereby common terms are reinforced and unique terms de-emphasised.
- Services provide an interface to functionality offered by devices, which can be discovered, composed and used by other devices within the environment. This requires mechanisms that enable a device to determine what services, offered by other devices, it can use. Services need to be discovered based on their capabilities and compositions need to be formed by processing and using service interfaces that match required service capabilities. Typically service interfaces describe the operations the service supports including the parameters (and their associated data types) they take and the values they return. Devices need to automatically process these signatures and determine if they can be composed with signatures supported by the devices local services.
- Devices must self-adapt to extend the functions they provide beyond what they were initially designed to do. They must also detect and rectify any conflicts as and when they occur within device configurations. Devices will automatically form relationships with each other based on what services devices provide and what services devices require. In this instance devices and/or services will connect too and disconnect from the network over time potentially rendering the composite solution incomplete. If a device or service is lost, an alternative must be found automatically with minimum disruption.

1.9 Novel Contributions to Knowledge

This thesis proposes a new framework we have developed for integrating networked appliances within device and service-rich environments so that high-level applications can be automatically created. Our proposed framework provides services that discover and

interconnect devices within the network; enable operational functions to be discovered and composed using semantic matching; select devices based on the capabilities they support; and allow device configurations to self-adapt to environmental changes. Each of the novel contributions we have made are discussed in turn in the following subsections.

1.9.1 Service-Oriented Networking

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 2003a, Mingkhwan 2004, Fergus 2005a, Mingkhwan 2005]. Each contribution we have made is listed below:

- Typical home appliances do not have the ability to provide their functions as independent services that can be utilised, simultaneously, by other devices within the environment. We have developed mechanisms to achieve this that allow devices to dynamically integrate themselves within any environment and disperse the functions they provide as independent services. Services may be pre-determined (middleware services that comprise our framework) as well as application specific (services wrapped around operational functions provided by devices) [Fergus 2003a], which can be simultaneously discovered and used by other devices within the network [Mingkhwan 2004, Mingkhwan 2005].
- Devices are manually connected and configured to work together in current home environments. It is becoming increasingly more complex to manage this process and therefore alternative mechanisms need to be developed to automate this. We have developed mechanisms within our framework that help achieve this that allow devices and services to be more accurately matched and integrated [Fergus 2005a].

1.9.2 Service Discovery

It is envisioned that application development will encompass the principles of service-oriented computing. As such it is important mechanisms are developed to accurately discover appropriate services. Current techniques are reliant on attribute-value pair matching, which is inherently restrictive since no universal taxonomy exists to describe services homogeneously. We have developed mechanisms that discover services based on semantic metadata that

describe what services do and what devices require [Fergus 2003a, Fergus 2003b, Fergus 2003c, Fergus 2005a]. Our novel contributions are listed below:

- Composing services in current implementations is based on carefully choreographed workflows or manual configuration. These approaches are inflexible and are difficult to implement in ad hoc environments. We have overcome this limitation by providing mechanisms within our framework that allow services to be described and discovered based on semantic metadata. This allows devices to dynamically discover, compose and execute services based on peer collaborations, devoid of any human intervention [Fergus 2003a, Fergus 2005a].
- As discussed, current implementations describe services using attribute-value pairs. This means that successful matches are only found if the service request exactly matches the service 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 services because of syntactic differences. In our framework we provide mechanisms that serialise service descriptions using high-level semantics that provide rich conceptual information about the individual functions devices provide [Fergus 2003b, Fergus 2003c]. Even if service requests and service descriptions are syntactically distinct but semantically equivalent our framework can find a match.
- It is difficult to get different device manufacturers to create and use a single standard for the terminology used to describe services. Consequently our framework uses high-level semantics to resolve the inherent ambiguities between service requests and service descriptions [Fergus 2003b].
- Applications that use semantic metadata rely on centralised knowledge sources managed by a consortium of knowledge engineers. Embedding heterogeneous devices within ad hoc environments makes it difficult to implement any kind of centralised solution. Devices need to host and manage their own knowledge, as such mechanisms need to be developed that allow devices to share and maintain this knowledge over time. In our framework semantic metadata resides on individual devices and the total knowledge within the network is the sum of all devices and their associated semantic information. No centralised servers are used to store this information, thus semantic information is distributed within the network, which ensures flexibility, fault-tolerance and fair concept creation and evolution [Fergus 2003b].
- Distributing knowledge within an ad hoc network makes it difficult to determine what knowledge is correct. Typically the consortium determines this however this is difficult when knowledge is embedded within devices that may not have a user interface. As such our base assumption is that knowledge needs to be managed

without any human intervention. Our framework allows semantic information to be dynamically evolved devoid of any centralisation using general consensus. Concepts that are more commonly represented are emphasised whilst less common concepts are removed from the network over time. This is an automated process that requires no human intervention [Fergus 2003b].

1.9.3 Device Capability Matching

One of the main features with service-oriented architectures is that functionality can redundantly co-exist. The difficulty is selecting the best service that meets the required configuration requirements. It may be acceptable to stream DVD content to a plasma TV, however the same is not true when a mobile phone is being used. As such service compositions must be based on the capabilities individual devices have [Mingkhwan 2004, Mingkhwan 2005]. The novel contributions we have made in addressing these challenges are listed below:

- Current service-oriented architectures rely on the user to determine which service(s) to select. The user determines what the best configuration should be in order to provide the best solution. Although this may not be too taxing on the user this is set to become increasingly more complex as networked appliances and home networks become common place. We have developed mechanisms that allow devices to automatically determine which device is better equipped to execute a given service [Mingkhwan 2004, Mingkhwan 2005]. This helps devices dynamically compose to create the solutions that provide the best quality of service.
- Existing capability specifications provide base solutions for describing device capabilities however they do not provide any quantitative mechanisms to make accurate comparisons. In our framework we extend existing specifications to include capability scoring which not only assesses individual device capabilities but also provides overall capability scores that assess the device as a whole. So even if a device is weak in one particular area, its overall capability score may still infer that it is the best device to use [Mingkhwan 2004, Mingkhwan 2005].

1.9.4 Dynamic service composition and self-adaptation

At present it is possible to implement networked appliances, however configuring and managing such an environment is problematic. It is becoming increasing more difficult for IT specialists and home users alike to install and configure next generation solutions. Consequently the base premise must be to target users with limited or no technical experience. As such mechanisms need to be developed that remove as much burden from the user as possible. Devices need to automatically integrate themselves within the environment

and manage themselves over time. Our framework provides several mechanisms that allow devices to automatically connect to each other to create high-level applications. Application solutions are managed by devices in compositions using our framework ensuring a given configuration is maintained [Fergus 2005a]. Again each novel contribution is listed below:

- Current middleware solutions provide mechanisms to disperse devices and services within the network however they do not provide any mechanisms that allow device configurations to automatically emerge. Device configurations are manually created by the user and thereafter managed. Again as we have argued above, as networked appliances and their associated configurations become more complex so will the integration and management tasks. This process needs to be automated. In our framework mechanisms are provided that allow devices to automatically form compositions with other devices to produce value added functions and aid zero-configuration [Fergus 2005a].
- Existing approaches do not provide mechanisms to detect conflicts and change configurations accordingly. Our framework allows devices to self-adapt to environmental changes as and when devices or services become unavailable to ensure that device compositions are maintained [Fergus 2005a].
- In existing approaches devices are interconnected, more often than not using wired solutions, by the user. Again the tasks associated with this are set to become increasingly more complex. Our framework provides mechanisms that allow relationships between devices to be automatically created to create high-level applications. This ensures that the user's defined quality of service is either surpassed or maintained [Fergus 2005a].

1.9.5 Ubiquitous Computing

Conventional computing is said to change as we see technology becoming more entwined within the fabric of our surrounding environment. However, current approaches favour enterprise solutions which exclude smaller devices with limited capabilities. By utilising service-oriented computing our framework avoids this restriction by allowing operational functions to be dispersed within the network. Our framework provides minimal functions that allow any device to be connected to the network irrespective of their capabilities. Any remaining functions the device is not capable of implementing can be discovered and used remotely within the network. We have made several novel contributions, which again we have published in [Fergus 2004, Fergus 2005b].

- Some devices, such as sensors will have limited capabilities and as such middleware solutions need to accommodate this. Many existing approaches fail to provide

mechanisms to achieve this, consequently such devices are excluded. Our framework can be implemented on devices with limited capabilities, for example sensors in a sensor network, which allows devices to be controlled using biofeedback [Fergus 2004][Bianchi 2003].

- Our framework allows the operational functions provided by devices to be dispersed within networked environments, which harnesses the power of wireless and mobile technologies, thus reducing the wires and cables that are part and parcel of all modern day appliances [Fergus 2005b].

These novel contributions extend current advances in networked appliance and home networking research initiatives and have helped create a framework that is highly flexible, extensible and self-adaptive. Our framework moves us closer to seamlessly interconnecting devices and realising zero-configuration. Several open standards have been enhanced to provide additional functionality that surpasses the functions these standards provide. These extensions fit more efficiently within new and emerging intelligent network architectures to embrace ubiquitous and pervasive computing environments. Furthermore, our framework provides highly adaptive mechanisms that allow any device, irrespective of its capabilities, to function within the network and decide how the framework services are used.

1.10 Thesis Structure

Chapter 1 of this thesis provides an overview of the problem domain, namely the inefficiencies associated with current networked appliances and home networking approaches. It highlights that little work has been carried out within ad hoc home network environments, and mechanisms for enabling devices and the services they provide to automatically form relationships. This Chapter argues that device integration and the management of device configurations needs to be automated to free the user as much as possible from the inherent complexities this process incurs. In doing so the challenges are presented, which include service-oriented networking, service discovery, device capability matching, dynamic service composition, self-adaptation and ubiquitous computing. This Chapter also describes a framework we have 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 and related work within the field of networked appliances. This discussion defines the key concepts used within this thesis and describes the limitations associated with current approaches. This Chapter also discusses how networked appliances relate to home networking and describes current middleware solutions that aim to interconnect devices within home environments. A discussion is presented

regarding how this integration is being performed using peer-to-peer (P2P) techniques, where several P2P models are presented. Each P2P model is discussed in terms of their associated functions, merits and limitations and an argument is presented regarding how P2P techniques can be used to loosely connect devices within ad hoc network environments. In this Chapter we also describe how techniques used within the Semantic Web and ontology engineering domains can be adopted to address several limitations within current service-oriented middleware architectures. The discussion argues that current service discovery mechanisms are inherently restrictive given that they are based on proprietary descriptions that dictate how services must be described and discovered, thus ignoring the semantics of information and the inherent vocabulary differences. As such an argument is presented pertaining to the use of semantics to better describe what services devices provide and what they require.

A detailed discussion of our new framework is presented in Chapter 3 and the core module each device must implement is presented. This Chapter includes the design models for the framework functions needed to connect the device to the network and communicate with other devices within the environment. A detailed design is presented using UML, which describes each of the design decisions made.

Chapter 4 is a continuation of Chapter 3, and describes in detail the UML design for all the remaining secondary services that comprise our framework. This Chapter describes the secondary services that do not need to be explicitly implemented by every device. The discussion focuses on the services used to perform semantic interoperability and ontology management; device capability matching; semantic service matching; and device self-adaptation.

In Chapter 5, an Intelligent Home Environment case study is presented which describes how the new framework implementation can be used to automatically discover and compose devices and the services they provide within the home environment. The case study also describes how devices within the home environment self-adapt as and when configuration changes occur. Several other application scenarios are presented in this Chapter illustrating how flexible the new framework is and examples are presented indicating how the framework can be applied to other problem domains.

Chapter 6 presents a detailed discussion on how the new framework is implemented. This Chapter discusses the toolsets used and highlights their merits and shortcomings. It presents the specifications the framework conforms too and discuses the implementation details. This includes an explanation of which tools where used to address the key requirements within the framework, how they have been extended to include new functionality and what functions and tools where problematic.

An evaluation of the framework implementation is presented in Chapter 7. Within this Chapter the framework and each of the secondary services and their associated functions are evaluated and discussed. The framework is also compared with existing middleware standards and each novel contribution made is discussed.

The thesis is concluded in Chapter 8, which provides a summary of each chapter and re-iterates the contributions made within this research project. Finally the future work is presented before concluding with some final remarks.

Chapter 2

2 Networked Appliances, P2P Networking and Semantics

2.1 Introduction

This section provides an overview of the work carried out in the main research areas relevant to this thesis, which includes networked appliances, home networking, peer-to-peer (P2P) technologies, and matching processable semantics. Cutting edge research initiatives are highlighted including their associated limitations, which are addressed within this thesis.

2.2 Networked Appliances

Devices are moving towards an increased reliance on interconnection. Games consoles, set-top boxes such as TIVO™ are extending the capabilities of conventional appliances to include networked communications. This provides the ability to play online games and tailor how and when we watch our favourite television programmes. Mundane tasks associated with general household maintenance such as vacuuming, security and mowing the lawn will be performed remotely by controlling devices using the Internet [Brooks 2002]. In this sense many devices of varied complexity will be a Web server. Researchers within the home automation industry believe that conventional household appliances such as the ones described above will form a major part of the future Internet as more and more devices become network-enabled.

There are several definitions of networked appliances, consequently it is difficult to provide a clear and decisive description of their key characteristics. From a hardware perspective, Moyer *et al.* [Moyer 2000] define networked appliances as "*a dedicated function consumer device with an embedded processor and a network connection*".

When trying to define networked appliances we also need to consider Internet appliances and make a distinction. Gillet *et al.* [Gillett 2000] explain that Internet appliances are the result of market pushes and consumer pulls. Mobile phones and Personal Digital Assistants (PDAs) have now become commonplace, whereby Internet access is either gained via the Wireless Application Protocol (WAP), Bluetooth and 802.11b wireless interfaces respectively. Consequently the intersection of functions provided by these devices leads to duplication. As a result, market and consumer demands are pressurising manufacturers to integrate these devices to create Internet appliances. Gillet *et al.* argue that although there is no clear

definition regarding what an Internet appliance is, a definition can be defined based on how such devices are marketed instead. They state that an Internet appliance is a consumer device that is not a PC; something that connects to the Internet; and something that does not make sense in a non-networked world. Driving such appliances is the need to reduce the complexity of PCs, which is being driven by three types of people; people with less disposable income; people who want to use the Internet, just not from a PC; and people who are happy using the PC, but want to extend the functions around the home [Gillett 2000, Gillett 2001]. In contrast a networked appliance differs from this definition, albeit it is a question of semantics, in that a networked appliance has a network interface, however it is not required to connect to the Internet – it could function perfectly well within a LAN. There is a fine line between these definitions, however the subtlety lies in the fact that a networked appliance could also be an Internet appliance (it could gain access to the Internet via its network connection, *i.e.* broadband), however an Internet appliance could not necessarily be a networked appliance, because it may only have the capabilities to connect to the Internet, but not interact within the local network.

Within our research we agree with the definitions presented above, however we place more emphasis on the software interfaces networked appliances provide. In this instance we therefore define networked appliances as devices that publish the functions they provide as independent services that can be discovered and used by other networked appliances in the network (LAN or Internet) to control, monitor, manage and extend the functionality they support beyond what they where initially designed to do.

2.3 Interconnecting Home Networked Appliances

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

2.3.1 Open Services Gateway Initiative (OSGi)

A well established middleware standard used to realise the digital home is the Open Services Gateway Initiative (OSGi) [OSGi Alliance 2005]. This standard has considerable industrial and academic backing from organisations that include Telcordia, Panasonic Technologies, Philips, Siemens and BMW. The alliance is composed of device manufacturers and service providers and its mission is to create open specifications for an end-to-end solution that enables the delivery of multiple services over Wide Area Networks (WANs) to home networks. OSGi was founded in 1999 by Alcatel, Cable and Wireless, Enron Communications, Ericsson, IBM, Lucent Technologies, Motorola, Nortel Networks and many more.

The framework incorporates three logically separated entities: the service and network provider, services gateway, and the in-home network. Service providers enable the provision of value added services to the residential customer via the services gateway. Whilst service operators, manage and maintain the services gateway and its services. Network providers offer the necessary network infrastructure to enable communications between the services gateway, the gateway operator, and the service provider.

Initially OSGi was designed as a mechanism to allow multimedia services to be provided within home networks via a set-top box. However as it has evolved the alliance has extended the capabilities of OSGi to surpass the functions provided by current set-box solutions. The services gateway protocol stack specifies standard APIs for the platform execution environment based on a Java Virtual Machine (JVM). The service framework itself sits on top of the JVM and provides a general purpose, secure, managed, service framework. Using the framework, applications known as bundles can be downloaded. Bundles are compressed Java archives files (Jar), which contain the resources to support the service (Java classes), including any dependency resources. Using Jar files for service deployment allows any service to be downloaded and controlled in a uniform way. The services gateway is controlled via a HTTP service on the gateway device. This service defines an API that allows service operators to configure the server as well as publish static and dynamic content. Access to the gateway is controlled using a device access service. This service allows service providers to communicate with and control devices connected to the home network, via the gateway. One of the important requirements from a user's perspective is to make the gateway transparent allowing users to view information in the gateway, modify its configuration, process notifications and interact with services. The configuration itself is performed using the Configuration Data Service, whilst the Persistent Data Service allows information generated by services to be stored. A generalisation of this service is the Logging Service, which allows monitoring data to be recorded pertaining to the gateway, the services and user interaction. The combination of these services forms the OSGi framework [OSGi Alliance 2005] and is a mechanism that allows devices within the home network to be accessed and controlled from external sources via the services gateway.

Configuring the OSGi framework is inherently human centric and in most cases managed and controlled via centralised service providers. Services are discovered and composed based on proprietary communication and middleware protocols. This is somewhat restrictive since distributed computing and service models are becoming increasingly more pervasive. As such devices and services are become more heterogeneous in nature. Consequently managing such a framework will be more complex. As technologies become more pervasive the amount of control placed on device and service integration becomes more difficult. Different device and

service providers will use different communication, middleware and service standards. As such interoperability is a problem that will require a more effective solution. New architectures need to be developed that overcome the restrictive proprietary nature of OSGi and provide a framework for more innovative solutions – the current OSGi standard does not have the ability to achieve this.

2.3.2 Digital Living Network Alliance (DLNA)

This in part has begun and research initiatives such as the Digital Living Network Alliance (DLNA) [DLNA 2004] formally known as the Digital Home Working Group (DHWG) [DHWG 2003] are developing interoperability standards. DLNA is also currently being used to realise the Intel Digital Home implementation [Intel 2003]. The primary goal of DLNA is to provide a framework that enables interoperability between devices that reside within three domains currently in existence within the home – these being the Internet, broadcast and mobile domains. They argue that consumers want the devices they own to work together within these domains.

DLNA advocates that the key to successful integration is to address customer demands where the devices they own work together within and across these domains. In order to achieve this, products designed for the home should be easy to install, must provide value, be cheap to purchase and interoperate with all other devices within the home. From a technical perspective DLNA argue that this requires design choices constrained through industry consensus that enable better interoperability. Currently open standards are too flexible and consequently interoperability between different vendors fails. However, such standards in conjunction with proprietary manufacturing are used because this is somewhat easier and in most cases reduces the time taken to deliver the product to high-street stores. The downside however is that such products have no effect on solving the interoperability problem.

The primary focus of DLNA is to move away from proprietary manufacturing and create a framework that interconnects the Internet, broadcast and mobile domains. The framework is based on a common approach which focuses on three key elements; industrial collaboration, standards-based interoperability frameworks and compelling products. From an industry perspective many Consumer Electronics (CE), mobile and PC industries have developed innovative consumer products, however this has been achieved very much independently of each other. No one single technology has the ability to guide interoperability alone.

This said each industry has made complementary contributions and offers unique capabilities and attributes. DLNA aims to incorporate these contributions into a standard that addresses interoperability. Through collaboration, standards form the basis for the creation of design guidelines that enable device manufacturers to develop devices that support a common

baseline for the set of required standards used. Standards developed by the consortium are not one-shot solutions but are continually evolved to support technological advances and the emergence of new and improved standards, where interoperability is the main driver.

Building on this vision, the current version of the DLNA framework addresses several key interoperability requirements. The building blocks include:

- Transparent connectivity between devices
- A unified framework for device discovery, configuration and control
- Interoperable media formats and streaming protocols
- An interoperable media management and control framework
- Compatible quality of service mechanisms
- Compatible authentication and authorisation mechanisms for users and devices

A number of design decisions have been made in the current specification and several existing standards are used. At the physical network layer wired and IEEE 802.11 wireless standards [IEEE Standards Association 2005] are supported using the IP network protocol. In the current specification this is based on IPv4, however future versions will include IPv6. Device discovery and control is achieved using Universal Plug and Play (UPnP) [Microsoft Corp. 2003], which is described below. The media transport protocol used is HTTP and several media formats are supported, which fall into two categories, required and optional. The required formats are JPEG, LPCM, MPEG2 and the optional formats are PNG, GIF, TIFF, MP3, WMA9, AC-3, AAC, ATRAC3plus, MPEG1, MPEG4 and WMV9. In the current version Digital Rights Management (DRM) and Content Protection (CP) are still under consideration.

The consortium aims to address interoperability and their base assumption is interoperability using agreed standards. Although it is not impossible it is not clear whether a single standard is capable of addressing all interoperability issues. The goal must be to utilise existing open standards as much as possible and interoperability mechanisms should be developed that abstract the underlying implementation details allowing any standard to be used and seamlessly integrated.

DLNA incorporates OSGi and as such it inherits the limitations associated with OSGi as described above. It is not clear how DLNA proposes to address the complexities associated with highly pervasive ad hoc environments. DLNA provides a base solution that is proprietary in nature; however it is not clear how scalable or flexible their architecture is.

2.3.3 Universal Plug and Play (UPnP)

A further standard that also has considerable industrial and academic support is Universal Plug and Play (UPnP) [Microsoft Corp. 2003]. This standard is in fact used by DLNA [DLNA 2004] to discover and control devices within the network. This standard is somewhat simpler than DLNA and OSGi because its sole purpose is to automatically interconnect, discover and control devices within the local home network. UPnP is a higher-layer protocol stack that aims to extend the simplicity of auto-configuration features of device Plug and Play (PnP) to the entire network enabling discovery and control of networked devices and services. UPnP is built on top of existing standards such as IP, HTTP and XML, which are used to enable devices to join the network dynamically, convey its own capabilities and learn the capabilities of other devices connected to the network.

The Home API working group and UPnP merged in 1999 to unify specifications for the development of home-control software. The specifications define an open network architecture based on well-defined principles, protocols and applications currently used in Local Area Networks (LANs). By utilising the benefits of the IP protocol, UPnP can be used over a number of physical media, which includes radio frequency (RF, 802.11x), phone line, power line, coaxial, IrDA, Ethernet, and IEEE 1394 (Firewire) [Poltavets 2005]. Consequently any medium used to connect two devices together can be used to implement UPnP. The UPnP standard is flexible and, although it is IP based, other technologies such as the Home Audio/Video Interoperability (HAVi) specification [HAVI 2003], CEBus and their associated Home Plug and Play (HPnP) standard [CEBus 2005], LonWorks [Chemishkian 2002] and X10 as demonstrated in the FP5 6Power project [Palet 2004a, Palet 2004b], can be used using UPnP bridges, proxies or residential gateways. For example, OSGi is often used in conjunction with UPnP. In this instance UPnP allows devices to be discovered and controlled within the LAN, whereas OSGi allows devices to be accessed and controlled via external sources.

The UPnP specification is comprised of four local node categories. Nodes can be control points, which are UPnP devices containing a set of software modules used to communicate with and supervise controlled devices. For example, a PC, PDA or set-top box may act as a control point. Controlled devices are less intelligent than control points. They are passive in nature and typically respond to control point commands and perform specific actions. A DVD or a VCR could be a controlled device. The UPnP working group realise that the specification will be used in conjunction with new and existing standards and as such the specification defines a UPnP bridge. This is a multi-protocol, multi-technology UPnP device that allows the UPnP network to be bridged with other technologies such as HAVi [HAVI 2003] and X10 as well as legacy devices. Such bridges may be requested if some devices are not UPnP

compliant, they do not have sufficient hardware resources or because the underlying communications medium does not support TCP or HTTP protocols.

UPnP [Microsoft Corp. 2003] achieves interoperability by leveraging existing mature standard protocols currently used on the Internet and LANs. A decision to use IP was adopted because it is seen as the de facto standard and has the ability to span different physical media allowing mature protocols like TCP, UDP, HTTP, DHCP and DNS to be used [Dean 2005]. It provides flexible mechanisms that can either use existing addressing schemes such as DHCP or AutoIP functions best suited to simple ad hoc networks [Dean 2005]. Devices and the services they provide are discovered using the Simple Service Discovery Protocol (SSDP) [Microsoft Corp. 2003], which enables home-network clients to discover networked resources. SSDP allows devices to announce their existence and for control points to locate the resources on the network. SSDP also allows devices to leave the network gracefully taking its services with it. The Generic Event Notification Architecture (GENA) [Microsoft Corp. 2003] is used for eventing. This mechanism allows devices to send and receive notifications to subscriber entities using the HTTP protocol over TCP/IP and UDP. Typically control points subscribe to event sources – GENA creates presence announcements which are sent to registered control points using SSDP. Any changes that occur with service states are also reported using GENA. Controlling the services provided by devices is achieved using the Simple Object Access Protocol (SOAP) [W3C 2005]. SOAP defines the use of XML and HTTP to execute services over the network using a form of remote procedure call (RPC). Using the existing standards defined above coupled with the UPnP specification protocols, UPnP defines a mechanism that allows devices and services to be discovered and controlled within local area networks.

The main limitation associated with UPnP is that it is human centric and does not provide any mechanisms that allow devices to automatically discover and compose devices and services without any human intervention. Discovery is based on attribute-value pair matching, which is restrictive and a poor mechanism for accurate device and service discovery. Compositions are carefully choreographed and control is based on application specific serialisations, *i.e.* predetermined SOAP messages. Furthermore devices can only be used that conform to the specification. This is somewhat restrictive and may isolate a large number of other networked appliances using different standards. Consequently the current version of UPnP, on its own, only provides controlled interoperability which is restrictive and again leaves little room for innovation.

2.3.4 Home Audio/Video Interoperability (HAVi)

Taking a more focused approach to interoperability is the Home Audio/Video Interoperability (HAVi) specification [HAVI 2003]. The HAVi architecture is a set of APIs, defined by a consortium of audio-visual electronics manufacturers who have developed a common, openly-licensable specification for networking digital home entertainment systems. HAVi uses a dedicated network based on the IEEE1394 standard [Poltavets 2005], which has a bandwidth capability up to 800 Mb/s. Such bandwidth capabilities enable isochronous communication and can simultaneously accommodate multiple real-time digital AV streams. HAVi facilitates multi-vendor interoperability between consumer electronics and computing devices and simplifies the development of distributed applications on home networks [Lea 2000, Nikolova 2003].

The HAVi architecture strikes a balance between the demands of consumers and vendors by facilitating both device interoperability and the introduction of new features or refinements. A key feature of HAVi is that each physical device has an associated software proxy. Adding new proxies to a home system makes new features or devices accessible even to applications running on older devices.

The software elements that comprise HAVi include the 1394 Communication Media Manager, Messaging System, Registry, Event Manager, Stream Manager, Resource Manager, Device Control Module, Functional Component Module, Device Control Module Manager and Applications.

HAVi supports inter-relationships between other networking standards; however this is from an audio/video perspective. The HAVi consortium sees this as an important aspect and aims to build bridges to offer additional consumer benefits. Using the HAVi specifications, the software API and the HAVi bridges, consumer electronics manufacturers can allow audio/video devices to operate within and across different networks irrespective of the underlying hardware or implementation details. This specification is designed to address interoperability and plug-n-play capabilities for audio and video systems; consequently this is a specialised standard that does not address wider interoperability issues.

2.3.5 Versatile Home Network (VHN)

Another home networking architecture is the Versatile Home Network (VHN) [CEA 2000, Ungar 2000] [Zahariadis 2003]. It was started in 1995 as the Video Electronics Standards Association (VESA) [Chen-Mie 1995, VESA 2005] Home Network. It was later transferred to the Consumer Electronics Association (CEA) and standardised by EIA as the (EIA/CEA-851) standard that defines a home intranet. VHN ties together home LANs, such as Ethernet

or IEEE 802.11a, allowing any device on a home network to communicate with any other device. The VHN architecture implements a whole home backbone, using IEEE 1394b, a long distance version of IEEE 1394a (FireWire). Local area networks, such as Ethernet or IEEE 1394a, connect to the backbone in each room, and IP is used to tie everything together. Version 2 of the VHN standard, was designed to incorporate UPnP for device discovery and control, SIP (Session Initiation Protocol)-based telephony [IETF 2004], network management, and security. It is compatible with OSGi [OSGi Alliance 2005] and HAVi [Williams 2001]. Another project that has adopted the VHN architecture is that of the Home Electronic System (HES) standard [ISO/IEC 2001]. This project attempts to define an architecture to standardise the use of available standards and protocols across the whole OSI layers from the physical layer to software applications [HES 2005].

The VHN architecture encompasses several existing home and middleware standards, such as UPnP, OSGi and HAVi, which have several limitations. As such the problems described above are evident within VHN. This architecture does not provide mechanisms for automatic service discovery and composition. Like other middleware standards VHN interoperability is carefully configured when the backbone is implemented. This requires high maintenance costs and lacks scalability. Each new standard used within the home must be carefully integrated into the VHN backbone. Mechanisms need to be developed that perform this process automatically. Devices must automatically adapt and integrate themselves within the environment irrespective of the underlying communication or middleware protocol being used. Again this requires a level of abstraction that hides the underlying implementation details. To date the VHN architecture does not provide any mechanism to achieve this.

2.3.6 Power Line Communication (PLC)

Matsushita Electric Industrial Co., Ltd (Panasonic), Mitsubishi Electric Corporation and Sony Corporation have joined forces to create a new alliance to define a new high-speed power line communication (PLC) standard. The consortium, aim to provide an interface standard between different devices, using electrical power lines for audio, video and data networking. This new alliance is called the Consumer Electronics Powerline Communication Alliance (CEPCA) [CEPCA 2005] and will promote PLC home networking worldwide by convincing CE manufacturers and the Information Technology sector to collaborate with device interoperability over power lines as the driving force.

The consortium believe that bi-directional PLC is a communication channel capable of supporting home networking using existing electrical power lines installed in home environments, which will enable high-definition video transmissions and the use of IP telephony. Through the consortium and the PLC-based standards it defines, interoperability

can be addressed between devices provided by different device manufacturers. Through the combined efforts of the consortium members, common standards will be developed for different PLC-based products.

Again like DLNA interoperability is addressed through common standards. As previously stated this is in theory possible, however in practice creating one single standard to address all interoperability issues is difficult. The PLC standard like many other interoperability standards is inflexible and requires carefully developed solutions. The cost of maintaining such solutions will be expensive and again restricts true innovation.

2.3.7 ePerSpace

Globally there are a number of research initiatives that are trying to address key requirements for next generation networked appliances and home networking. The ePerSpace [France Telecom 2005] project aims to develop an end-to-end solution for personalised value-added audiovisual services contained within the home and external environments that will increase user acceptability of such systems. ePerSpace provides distributed multimedia services which are accessed via an open access network (OAN) based on the details defined in personalisation profiles that allow content and user devices to be dynamically adapted to specific users. The approach taken by ePerSpace is to create a trusted and interoperable integrated framework to seamlessly interconnect heterogeneous audio and visual devices. This also includes home platforms that define generic business models for mass-market adoption. This framework aims to address interoperability problems and the management of service platforms including service and context adaptation using personalised data.

The ePerSpace framework provides Global Network Integration and Interoperability mechanisms that allow audio and video content to be transmitted between distributed services using secure shared user profiles. Through this framework environments are dynamically built to include networked appliances that can be controlled by content creators using Rich Media Object Management tools. Currently, aspects of the ePerSpace research initiative are being used by the BT Extract project on consumer vehicle telematics [Millar 2004], investigating the continuity of home-car services, with a particular focus on personalisation.

This standard attempts to move us one step further than the standards described above to add a level of “intelligence” that provides context adaptation mechanisms based on user profiles. However, again this is a carefully choreographed solution, based on proprietary standards that will be difficult to implement in pervasive ad hoc environments. Contexts are serialised using common standards and context adaptation is achieved by reasoning over these standards. This solution assumes a close-world view and as such maintaining and managing this solution is costly. New standards, devices or services integrated within the environment have to either

conform to the ePerSpace specification or adaptation mechanisms need to be developed that integrate new device and service types. It is not clear at this stage how this can be achieved. Although ePerSpace talks about adaptation this appears to only be between predetermined profiles. Adaptation must filter down to the device and service layer whereby automatic device and service compositions self-adapt based on environmental changes. The ePerSpace literature does not suggest that this is the case.

2.3.8 MediaNet

MediaNet [Travert 2004] also aims to develop an end-to-end solution for multimedia content distribution. The project aims to create a framework that provides multimedia communications for content distribution services for residential markets. The framework takes into account the complete supply chain to manage the collaboration between content owners, network providers and middleware services.

The underlying principle adopted by MediaNet is to provide an open architecture that provides common access mechanisms for interworking home networking platforms. The open architecture is achieved using pre-defined standards, common interfaces and well understood business models. The framework will provide mechanisms that allow content to be distributed and accessed, interworking, multimedia content to be stored, digital rights management and high-quality audio and video distribution between wired and wireless devices. Application developers, service providers and equipment manufactures can use MediaNet to implement new applications compatible with common infrastructures and interfaces, including networked devices.

MediaNet extends existing In-Home networking technologies to include In-Home management that enables interoperation between services provided by external service providers and In-Home application services and also provides mechanisms for deploying and controlling networked services in a user-friendly way. MediaNet is currently researching how this can be achieved using existing standards like OSGi [OSGi Alliance 2005] and UPnP [Microsoft Corp. 2005]. As such MediaNet also experiences the same limitations described for OSGi and UPnP above. It is not clear from the literature whether MediaNet aims to address these issues. However the interoperability standards being developed for multimedia content could be integrated into different interoperability middleware solutions to solve specific interoperability problems.

2.3.9 RUNES

As well as multimedia content, other research initiatives are concerned with the actual internal and external control of household appliances. Playing a key role in this will be sensor

networks, which are said to become entwined within the fabric of home environments. One such project investigating this is the European funded Reconfigurable Ubiquitous Networked Embedded Systems (RUNES) [Koumpis 2005] project. RUNES claims that embedded systems and the Internet will begin to merge to create truly pervasive networked computer systems. This combination will result in complexity due to heterogeneity and the dynamic nature associated with networks that resist any form of control. In spite of this, there is a need to promote this integration because bespoke development is too expensive and too limiting for innovative applications.

The RUNES project aims to address this complexity using a scalable middleware framework including application development tools that will allow users, designers and programmers the flexibility to interact with services, devices and sensors and ease the overall application development process. This framework claims to be adaptive, robust and self-organising. The project is in its early stages and it is not clear whether a middleware architecture can be created to enable the creation of a large-scale, distributed, heterogeneous network system that can seamlessly interoperate and dynamically adapt to environment changes.

2.3.10 Semantic HiFi

A new area of research, seen as a key enabling technology within home networking, is the ability to effectively describe and discover multimedia services using ontological structures. The Semantic HiFi [Jacob 2004] project falls under this category and aims to address the limitations associated with attribute-based audio processing. The Semantic HiFi framework allows users to discover music stored on a particular device or on another device that may reside within the home network or the Internet.

Semantic HiFi uses a peer-to-peer network to distribute and discover music and meta-data provided by home users, music labels, and amateur musicians. The framework provides a set of libraries, semantic description schemes, specifications and guidelines that enable interoperability between different applications. Each Semantic HiFi application contains a metadata repository which is used to store audio fingerprints including metadata for individual tracks, which are shared within the peer-to-peer network.

Semantic HiFi supplements semantic descriptions to include hash functions and audio fingerprinting to standardise how files and musical content are identified. This provides a more robust identification mechanism which is independent of the file type, audio encoding, amplitude, and silence header. Applications use audio fingerprints to query the Semantic HiFi network for metadata. The metadata itself is standardised in order to ensure interoperability between metadata descriptions used by other devices within the network. This project addresses an important requirement and as we see a large number of services and multimedia

content becoming common place within home networking platforms, selecting the correct content will be paramount and a key factor for user acceptance.

2.3.11 Future Home

Connecting appliances using a wired building infrastructure is far more expensive and may only be available for new buildings. In a typical ubiquitous environment, users need their complex networked appliances to be capable of communicating anytime and anywhere, and more significantly this must be done seamlessly and wirelessly. A wireless connection does not need any rewiring and the full system can be up and running within minutes. The European funded Future Home Project [Future Home 2005] is trying to address this issue by creating a solid, secure, user friendly home networking concept with an open, wireless networking specification. The project uses IPv6 and Mobile IP protocols in the wireless home network. It also uses a generic device interface to make it easy and cost effective to insert intelligence and communication capabilities in home appliances.

The ability to monitor and control appliances and consumer electronics remotely has interested users for decades. Whether it is through mobile and land-based phones, digital keypads or over the Internet via Web and WAP interactive sites, mobile users are becoming more demanding in terms of monitoring and controlling the status of their homes and their appliances. The HomeOnAir project [Barba 2005] has proposed the provision of advanced home control services using wireless remote access based on WAP technology. It provides a description of the services, architecture and human-machine interfaces and provides a complete HomeOnAir system that is available for installation. A platform, that can manage Lonworks and X-10 home automation networks, has also been provided.

Researchers are also looking at how existing technologies can be used to realise different applications. This is becoming more popular in the area of patient care within residential homes. For example, extending the concept of peer-to-peer chat programs homes can be equipped with bi-directional communications between health centres and patients to perform on-demand care. Furthermore, utilising advances within biofeedback, appliances can be controlled and information can be sent to medical practitioners who could then interact with the patient and the home to control networked appliances.

A project investigating this is the HomeTalk project [HomeTalk 2005]. This project has proposed a voice-enabled, residential automation and networking platform to allow the capability of communicating with the residents via a natural voice interface. It creates technology for a human-centric, fully automated home with built-in intelligence and natural language capabilities. The full implementation proposes to embed the voice interface capability in the residential gateway/controller (RG) and support local interaction via any

indoor/outdoor network through ordinary telephone lines, wireless microphones or emerging voice-over-broadband and the Internet.

2.3.12 WCAM

Similarly, the WCAM project [Meessen 2004] is an initiative to develop a system for audio-visual content delivery over a wireless, seamless and secured network by exploiting the technology convergence between video surveillance and multimedia streaming over the Internet. It proposes an integrated solution for smart delivery of video surveillance data. This includes smart video coding based on automatic scene analysis and understanding. Specifically, the segmentation results are used for encoding regions of interest (ROI) in Motion JPEG 2000 guaranteeing good quality for the semantically relevant objects while keeping a low average data rate. By linking image analysis, such as segmentation and object tracking for both vehicles and people to the video encoding the method is proposing to reference images and segmentation using shape, colour or texture analysis. This process will output active frames and ROI that need to be encoded with better quality and described by means of metadata. The video content can also be secured using a Digital Rights Management (DRM) system and privacy issues are addressed by selective protection of sensitive frame regions.

2.3.13 BETSY

Wireless multimedia streaming on handheld, mobile or other battery-operated devices is a major technology underlying the next generation information and entertainment appliances. Today it is not possible, even at design time, to make well-founded system trade-offs between network and terminal resource consumption, energy consumption of the terminal and timeliness of the streaming data. The BETSY [BETSY 2005] project is aiming to deliver the theory, models and design methodologies to make this possible during design time. It is also devising a framework implementation that makes dynamic adaptations, in this trade-off, possible at run-time. The project proposes to combine the research results of several domains, such as networking, device resource management, real-time processing and stream processing, to achieve a holistic view of the dependencies between bandwidth, delay, schedules, and the power and energy consumption for this specific application domain. The aim is that the results will lead to reduced product cost by eliminating pessimistic and large safety margins or improved system performance with equal resource demands.

2.4 Peer to Peer Networking

Peer-to-peer (P2P) computing dates back to the first networks developed during early Internet research projects such as ARPANET. ARPANET was carried out by Bolt, Beranek and

Newman (BBN) Technologies [BBN 2004] and was funded by the Advanced Research Projects Agency (ARPA), which was changed to the Defence Advanced Research Projects Agency (DARPA) in March 1972 [DARPA 2003]. This was the first large-scale network to be developed and was based on packet-switching within a Wide Area Network (WAN). The early developers of ARPANET envisaged that computers would be connected throughout the world in a peer-to-peer fashion, whereby resources could be shared, thus the term peer-to-peer emerged. In fact the early Internet was a P2P network and every node had a permanent IP address.

In this model Computers were connected via a pre-determined communication protocol called the Interface Message Processor (IMP). The IMP acted as a digital interface on each computer and performed the functions of dial up, error checking, retransmission, routing and verification. Roberts [Roberts 1967] describes the combination of the telephone lines, the IMPs' and the data sets as the message switching network. The first IMP installation took place during 1969 and by the middle of 1972 there where twenty three connected computers, which were located in San Francisco, Utah, Michigan, Illinois, Pittsburgh, Boston, Washington and Los Angeles.

The ARPANET was decommissioned at the end of 1991 and was classed as the forerunner of today's Internet. Although ARPANET no longer exists in its original form, many of its parts have progressed into the current Internet, including the TCP/IP protocol [Murhammer 1998] – TCP/IP replaced the Network Control Program (NCP) protocol in 1978 [Murhammer 1998] – which was developed as part of the ARPNET project [Feibel 2000]. With the advent of the Internet and more recently the World Wide Web (WWW) [Berners-Lee 1989] the client-server model has become one of the most common business models for distributed computing and as the number of interconnected computers increased, so sparked the problem associated with the number of available IP addresses. It soon became clear, based on IPv4, that there was not enough IP addresses to accommodate every machine connected on the Internet. Consequently, it has become impossible to connect every device in a true P2P fashion whereby each device has its own IP address.

This problem has been addressed in part using the Network Address Translation (NAT) protocol, which allows public IP addresses (the range of addresses available under IPv4) to be mapped onto internal private IP addresses. Thus computers in the centre of the network are used as a means of connecting the organisation to the outside world, which themselves are connected to all computers in the internal network using private IP addresses. The process of allowing internal computers to communicate with the outside world, via the organisations public IP address, is achieved using NAT. Although efforts in IPv6 are well underway, this model still remains the dominant model to date.

P2P however is re-inventing itself and is once again becoming the distributed computing model of choice. Although P2P is still seen as a disruptive technology, industrial and academic institutions are beginning to view these networks as real enablers for new and innovative applications. These networks are scalable and highly adaptive and provide considerable benefits over current client-server solutions. As such, many P2P implementations exist today, and many more are being created. Many of these applications support their own proprietary protocols and P2P models, categorised as hybrid, pure, unstructured and structured. The P2P applications considered within this thesis are listed in Table 2.1.

Hybrid	*Pure*	*Unstructured*	*Structured*
Napster	Gnutella	Napster	Chord
JXTA		Gnutella	CAN
			Pastry

Table 2.1 P2P Models

Each P2P protocol listed in Table 2.1 is discussed in more detail in the following subsections.

2.4.1 Napster

One of the earliest P2P implementations that brought P2P computing to the forefront and which sparked a large amount of media attention was Napster [Oram 2001]. Napster was created purely for the distribution of MP3 audio files (an MPEG-1 Layer 3 audio encoding) [Brandenburg 1999], and as such it was swamped with negative press because people where downloading digital content illegally, subsequently ignoring content copyright. Each Napster node downloads and installs the client software used to connect the peer to the centralised Napster server. Once connected, peers share MP3 files stored locally on their hard drives, which are then indexed by the Napster server. Clients submit queries to the Napster servers for a particular audio file. This results in a list of files that match, which includes the connection information, username, IP and port address the querying client must use to connect to the peer that has the file. Once the querying peer has this information it attempts to connect to the peer and transfer the target content in a P2P fashion. At this point the Napster server is no longer required [Gradecki 2002].

Although Napster proved successful and is said to be the grandfather of modern P2P computing models it suffered from a number of limitations. The major limitation was the fact that it could only share MP3 content. The other limitation lay in the fact that it was a hybrid model reliant on client-server technology - if the server becomes unavailable then the discovery mechanism used to find content is lost. This marked the demise of Napster when it was ordered to switch off its servers in 2001.

2.4.2 iMesh

Another hybrid protocol, similar to Napster called iMesh [iMesh Inc 2005] uses a centralised server, which clients connect to and search for content. However the iMesh model differs somewhat to Napster in two main areas. Firstly it allows any content to be shared including MP3 audio files. Secondly, and the reason why iMesh has not been subjected to the same legal problems as Napster, it has a mechanism to remove copyrighted files from the network.

2.4.3 Gnutella

Computational expense and scalability issues associated with the above mentioned models are well documented, which has resulted in new P2P networks devoid of any centralisation. The most popular being the Gnutella protocol [Gnutella 2001]. Like iMesh is provides a generic file sharing mechanism that allows any digital media content to be shared. However it differs from iMesh and Napster because the Gnutella protocol uses a purely decentralised model, which is not reliant on any centralised authority. Another distinguishing feature is its use of the HTTP protocol to transfer information. In effect a Gnutella node is like a Web server.

The search mechanism used by Gnutella adopts a different approach to Napster in that it does not require any centralised server to manage the location of content within the network. Search packets are used with predefined TTL values, the default value being 7, which corresponds to the number of hops the message can take. The packet is passed to all the immediate peers' the querying peer is connected to, which in turn is passed to all the peers the peer is connected to. The Horizon as defined by Kan [Oram 2001], given a TTL of 7 encompasses about ten thousand nodes. If a node is found that contains the file, the information is routed back to the querying peer, which can then be downloaded directly from the target node.

Unlike Napster, it is difficult to disrupt the network because no one single node is responsible for creating it. If any given node is lost it does not affect the overall search mechanism of the Gnutella network. The worst case is that you only lose the content provided by that node. Consequently Gnutella provides mechanisms to counteract some of the limitations associated with Napster. As such many Gnutella clients have been developed since the protocol was first released in 2000, which include Bearshare [Free Peers 2005], Shareaza [Shareaza 2005] and Limewire [Lime Wire LLC 2005].

2.4.4 FastTrack

The FastTrack protocol claims to be better than Gnutella and its variants. Unlike Gnutella this protocol is proprietary, consequently specification details are difficult to find. A number of popular applications such as Kazaa [Morle 2003], Morpheus [StreamCast Networks 2005]

and Grokster [Grokster 2005], use the FastTrack protocol which divides users into two group types. The first group contains supernodes and the second contains ordinary nodes. Supernodes are defined as computers with significant computation, network and bandwidth capabilities. Supernodes are automatically selected, and typically owners do not know that there machines are acting as a supernode. All supernodes are connected together to create an overlay network that acts like a hub and processes all data requests received from ordinary nodes within the network, which are inherently less capable nodes. Each supernode may serve between 60 and 150 ordinary nodes at anyone time.

Initially when applications such as Kazaa are installed it uses pre-coded supernode addresses, which act as bootstrapping nodes. When Kazaa is started it is registered with the "central server" and chooses a supernode from a list of supernodes on that server. When a node wants to share or search for a file a request is submitted to the supernode, which in turn submits it to all other supernodes, which in turn propagate the request to the ordinary nodes it is servicing. Like Gnutella, messages are configured with a TTL value of 7, ensuring that message propagation is terminated once seven hops have been reached.

Once the content has been found it is transferred directly from the target node to the querying node using the HTTP protocol, without using the supernode. There is a subtle distinction between the FastTrack model and that of Napster in that the Napster server managed an index of audio file information, which includes information about the peer sharing the file. According to copyright laws this was deemed illegal and a copyright infringement even though the file did not physically reside on the Napster servers or even facilitate in the physical transportation of the file. The FastTrack protocol avoids this problem because it only manages a list of supernodes and not information regarding the content itself. Supernodes are ad hoc in nature and are free to join and leave the network at any time. So information about supernodes held by the FastTrack servers continually changes. This abstraction detaches the FastTrack protocol, including the applications that use the protocol, from media content and thus some believe that FastTrack-based applications do not aid copyright infringement.

2.4.5 Chord

P2P network topologies are typically defined as hybrids, such as Napster, which use both client-server and P2P techniques or pure as is the case with Gnutella. However further distinctions have emerged as P2P systems have evolved which classify P2P networks as unstructured (as is the case with Napster and Gnutella) or structured (as is the case with Distributed Hash Table (DHT) based P2P implementations such as Chord [Dabek 2001], CAN [Ratnasamy 2001] and Pastry [Rowstron 2001].)

Chord is a structured P2P network that allows order to emerge using its DHT routing algorithm. Its basic structure forms a ring topology, whereby each node only has to establish one connection. The protocol describes how peers join the ring, how data is stored and how the network deals with failures [Dabek 2001, Eberspacher 2004].

Chord uses a hashing function, such as SHA-1, to generate node and object identifiers known as keys. The node identifier is created using the IP address and port, whilst the object identifier, which can be any kind of shared content, is created using the data to be shared within the ring. Node identifiers are arranged in a circle modulo 2^m, where m is the length of the hash value. Every key k is assigned to the node whose identifier n is larger than or equal to the hash value of k. The node the key belongs to is called the successor. In Chord, node identifiers increase clockwise and keys are assigned to the first nodes that reside closest to them clockwise. In this instance Chord is a hashing function, designed to distribute keys evenly throughout the ring topology, whereby all nodes roughly receive the same number of keys.

Finding nodes that map to the key is performed with little routing. Every node is aware of their successor and as such queries are passed from successor to successor. When a node is reached that has a hash value bigger or equal to the hash value of the key, then a node has been found that can map the query to the key. Although this mechanism works, it would however be inefficient in large rings because every node needs to be traversed. Chord addresses this problem using a finger table. Each node has a finger table that is capable of indexing t entries, where t is the number of bits in the identifier – if SHA-1 is used this would be 160. Each entry of index i points to a node s that succeeds node n by at least 2^{i-1}. The node s is known as the i^{th} finger of node n. Using this mechanism the first finger within the table is always the nodes immediate successor.

In order to overcome the need to traverse every node, a node can use the entries contained in the finger table to try and find the predecessor of some key k. Node n achieves this by searching its finger table for some node x that immediately precedes some key k. If it finds node x then it queries the node to determine which node is closet to x. By repeating this process n moves the query closer and closer to k. In Chord this is called iterative routing.

As with any other P2P network, nodes will continually connect and disconnect from the ring. As such the successor and predecessor relationships between nodes and keys, including the finger tables will change. Chord addresses this problem using a stabilisation scheme designed to repair the ring when new nodes arrive and existing nodes leave. Each node periodically runs the stabilisation function to correct incorrect successor and predecessor entries. When node n runs the stabiliser it asks its successor s for its predecessor p. Under normal conditions

this will be *n*. However if a new node enters the ring and its hash value falls between the hash values for *n* and *s* then *n* has to update its successor entry to now point at the new node that has joined. The old successor used by *n* is notified about the change so that it can update its predecessor entry. Lastly the stabiliser notifies *n'*s successor, which is the newly added node, about its existence so that the new node can enter *n* as its predecessor. Although there are additional features supported by Chord, this overview describes the basic functionality [Dabek 2001, Eberspacher 2004].

2.4.6 Content-Addressable Network (CAN)

Another similar protocol to Chord is the Content-Addressable Network (CAN) protocol [Ratnasamy 2001] which uses the DHT concept. CAN comprises a number of nodes that form a overlay P2P network that store chunks, known as zones, of the hash table. Each node also contains information about the adjacent zones in the hash table. Requests, which may be *insert*, *lookup* and *delete*, for a particular key are routed towards the CAN node whose zone contains the key. Like, Chord, CAN is a decentralised P2P network, which requires no centralised server to index and discover content.

The central idea surrounding the CAN protocol is based on a virtual *d*-dimensional Cartesian coordinate space. The space is dynamically partitioned among all the nodes in the system. This means that every node owns its own zone within the global coordinate space. This space stores key-value pairs where k_1 is mapped onto a point *p* in the space using a uniform hashing function. The key-value pairs are stored on the node that owns the zone in which *p* resides. To discover the values of some key k_1 any node can use the hash function to map k_1 onto point *p* and retrieve the contents from *p*. This may be the content or a pointer to the content. If the point *p* is not owned by the querying node or its neighbour, then the request is routed towards the node where point *p* resides.

CAN nodes perform this type of routing using information about the zone and coordinate information of its neighbouring nodes. The neighbouring nodes in the space server have a coordinate routing table that allows information to be routed between any two nodes. Each node maintains its own routing table, which contains information about IP addresses and zone coordinates for all its neighbouring nodes. Two nodes are classed as neighbours if their coordinates overlay around *d-1* dimensions, *i.e.* in a two dimensional space two nodes are neighbours if either the X or Y coordinates share the same value. In this instance, node (0, 1) would be a neighbour of node (1, 1) because the Y coordinates for both nodes are the same. Messages sent within the CAN network contain the coordinates for the destination. Using its neighbours coordinate set, a node routes a message towards its destination using a mechanism

called greedy forwarding in CAN, which forwards messages to neighbouring nodes with coordinates closest to the destination coordinates.

There may be many routes that exist between any two nodes within the CAN network, consequently if neighbouring nodes fail or leave the network, the message can be routed along an alternative route. In more severe instances where all the neighbouring nodes fail and the repair mechanism has not rebuilt the mesh then greedy forwarding will temporarily fail. The CAN protocol makes provisions for such an eventuality using a technique called expanded search, which locates a node closer to the target node – when a node is found, the greedy forwarding mechanism continues.

New nodes can join and leave the CAN network over time, which dynamically changes the mesh configuration. In the instance when a new node joins it discovers an IP address of any node within the CAN network. No constraints are placed on how this is achieved, however bootstrap servers are used within CAN. Once a node has been found, the new node selects a random point *p* in the coordinate space and sends a JOIN message. The message is forwarded to the node whose zone contains point *p*. This node upon receiving the JOIN message splits its zone in half and assigns one half to the new node.

Once the new node receives its zone the node uses the IP addresses of its neighbours, whilst the previous owner of the zone updates its neighbour entries in its routing table – nodes that are no longer neighbours are purged. The old and new node neighbours are notified of the change, which results in each node updating its routing table. As well as update messages each node periodically sends refresh messages to its neighbours containing the node's current zone coordinates. The neighbours use these messages to update their routing table. The procedure described here is localised so that newly added nodes only affect nodes which are its direct neighbours. How many neighbours a node has is dependent on the dimensionality used in the coordinate space.

In the case where nodes leave the space, either voluntarily or because of node or network failure, zones are automatically reallocated. In controlled situations a node hands over its zone and its associate key-value pairs, to one of its neighbours who have the smallest zone. Conversely, there will be instances when a controlled handover is not possible, for example when the node suddenly fails. Using a takeover algorithm a neighbouring node takes over the zone. However the key-value pairs are lost until the state is refreshed by the holders of the data.

As mentioned earlier nodes send update messages. The prolonged absence of messages indicates that a node has failed. Once a neighbour determines that a node has failed it initiates a takeover procedure. The node with the smallest zone should take over the available zone.

Determining the neighbour with the smallest zone is achieved by each neighbour starting a timer – when the timer expires, a takeover message is sent to all the neighbours and when these messages are received the node cancels its timer if the zone size in the message is less than its own zone size. Alternatively the node responds with its own takeover message. This mechanism allows neighbouring nodes to determine which node has the smallest zone and thus provides a node selection mechanism to choose which node will perform the takeover. This section describes the basic functionality of the CAN protocol, however for a more detailed description including the enhancements to this protocol see [Ratnasamy 2001].

2.4.7 Pastry

The Pastry protocol is also similar to Chord and CAN, which is a self-organised overlay network of nodes, where each node routes client requests. Pastry nodes are identified in the network space using a 128 bit identifier, known as the *nodeId*. The *nodeId* indicates a node's position in the circular *nodeId* space. The *nodeId*s themselves are assigned randomly when the node first connects to the Pastry network. Several mechanisms can be used to derive the *nodeId*, however typical implementations use the nodes public key or IP address to create a hash. In Pastry *nodeId*s are thought of as a sequence of digits in base 2^b. Using this mechanism messages are routed towards the *nodeId* that is numerically closest to the message key. For example a node uses its routing table entries to forward the message to one of its neighbours whose *nodeId* shares with the key a prefix that is a least one digit longer than the prefix that the key shares with the present *nodeId*. If no such node is known, the message is forwarded to a node whose *nodeId* shares a prefix with the key that is as long as the current node, but is numerically closer to the key than the present node is.

Nodes within Pastry maintain their own routing table, which is organised into $128/2^b$ columns. For example, *b* could be 4 consequently there would be 8 rows and 16 columns. The 16 entries in row *n* contain the IP addresses of nodes whose *nodeId* share the first *n* digits with the present nodes *nodeId*. Furthermore the *nth + 1 nodeId* digit in the candidate *nodeId* has one of the 2^b possible values other than the *nth + 1* digit in the present *nodeId*. Entries in the routing table are left empty if no node with the appropriate *nodeId* suffix is known.

Determining the value of *b* is a trade-off between the size of the populated portion of the routing table and the maximum number of hops required to route a message between any two nodes. The size of the populated portion of the table is $log_{2b}N * (2^b - 1)$ where *b* is the base and *N* is the number of nodes. The number of hops required can be calculated as $log_{2b}N$.

As well as the routing table, each node also maintains a neighbourhood set *M*, which contains *nodeId*s and IP addresses of the *M* nodes that are closest to the local node. The set is not used for routing, but rather for maintaining locality properties [Rowstron 2001].

Nodes also maintain a leaf set *L* which contains a set of nodes with the numerically closest larger *nodeId*s and numerically smaller *nodeId*s, relative to the present nodes *nodeId*. The leaf set is used when messages are routed. When a node receives a message it first checks to see if the key falls within the range of *nodeId*s covered by its leaf set. If it is, the message is forwarded directly to the destination node. If the key is not covered by the leaf set, the routing table is used and a message is forwarded to the node that shares a common prefix with the key by at least one more digit. In certain cases, it is possible that the appropriate entry in a table is empty or the associated node is not reachable, in which case the message is forwarded to a node that shares a prefix at least as long as the current node and is numerically closer to the key than the current *nodeId*.

Pastry provides mechanisms to self-organise and adapt to network changes. In the case where a node arrives, it needs to initialise its state tables and inform other nodes of its presence. An assumption is made that the node knows about a nearby Pastry node *A*. This could be achieved using multicasting. The new node asks node *A* to route a special *join* message with a key equal to the new *nodeId*. Messages used to join a node to the Pastry network are like any other Pastry message, consequently Pastry routes the *join* message to a node *Z* whose *nodeId* is numerically closest to the new node. In response to the *join* request nodes *A*, *Z* and all nodes *en-route* send their state table to the new node, which are used to initialise the new node's state table. Lastly the new node informs any nodes of its arrival. This procedure ensures that the new node initialises its state with appropriate values, and that the state in all other affected nodes is updated [Rowstron 2001].

Nodes will depart and even fail over time without warning. In Pastry nodes can determine whether neighbouring nodes have failed if communication can no longer be established. A failed node in the leaf set is replaced by contacting its neighbour in the *nodeId* space and asking that node for its leaf set. Using this leaf set the current node updates its own leaf set to replace the failed node.

Failed routing table entries are repaired lazily, whenever a routing table entry is used to route a message. Pastry routes the message to another node with a numerically closer *nodeId*. If the downstream node has a routing table entry that matches the next digit of the message key, it automatically informs the upstream node of that entry.

If a numerically closer node can be found in the routing table, it must be an entry in the same row as the failed row node. If that node supplies a substitute entry for the failed node, its expected distance from the local node is therefore low, now all these nodes are part of the same nearby nodes with identical *nodeId* prefixes. If a replacement node is supplied to the downstream, a routing table maintenance mechanism is triggered to find a replacement entity.

DHT-based P2P implementations are said to provide considerable benefits over previous generations and provide emergent behaviours that support order and increased performance. There is however a trade off between performance and maintenance costs. For instance the cost associated with maintaining a consistent distributed index in DHT-based solutions is high because most time is spent updating indices. It is generally agreed that DHT provides an efficient mechanism for data access however costs are exponential as the number of peers that continually connect and disconnect increases. The converse of this problem is that not having a DHT requires an exhaustive traversal of the network, which results in network flooding. Using this technique removes the maintenance costs associated with keeping the network topology consistent, however it is penalised in terms of network congestion.

Whilst implementations like Chord, CAN and Pastry may work well in structured network environments like organisational P2P networks (where the network structure remains largely the same) they are not as effective in unstructured environments (as is the case with Gnutella and FastTrack). This is because these networks are inherently ad hoc in nature and highly unstructured. The network topology is continually changing and consequently managing a consistent DHT across such networks requires considerable effort.

2.4.8 JXTA

New P2P initiatives, more specifically JXTA (Juxtapose), have tried to create a balance by creating a hybrid system that uses a loosely consistent DHT [Traversat 2003]. JXTA in this sense is similar to other implementations such as Chord by virtue of using DHT. However the way in which a table is managed differs. Whilst Chord relies on more costly mechanisms to keep the network view consistent, JXTA uses a less costly mechanism that ensures the network view is only loosely-consistent. The advantage with this approach is that it is less expensive to maintain, however the disadvantage is that it may be temporarily or permanently inconsistent.

The JXTA architecture consists of three layers; the core layer; the services layer and the application layer. The core layer provides the main services required for P2P computing such as peer discovery, peer creation, groups, security and mechanisms for mobile devices, such as mobile phones and personal digital assistants (PDA) [Gong 2001, JXTA 2001, Qu 2001, Waterhouse 2001, Halepovic 2002, Oaks 2002, Traversat 2002, Wilson 2002, Arora 2003, Sun Microsystems Inc. 2005a, Sun Microsystems Inc. 2005c]. The service layer provides services that are deemed desirable, for example file sharing, protocol translation and authentication. The application layer contains any number of P2P applications, built on top of the services layer, to perform some given function, for example solutions provided by DLNA, OSGi or UPnP.

The JXTA protocols allow any device to discover and communicate with each other and provide mechanisms to perform interoperability between heterogeneous devices. Devices may implement JXTA in any programming language and form bindings with any underlying transport protocol on any platform.

Devices are known as peers in JXTA which are nodes that sit inside the network. Communications take place between peers, which may reside within and across different networks, by sending XML messages along communication channels called pipes. Peers are dynamic in nature and are free to connect and disconnect at any time. This behaviour means that peers dynamically reconfigure as network changes take place. Peers connect to form peer groups, which emerge through inter-peer connections, known as relationships. Peer groups are a logical grouping of peers that share a set of common services. Many peer groups may co-exist, which can be identified using globally unique IDs. Peers are free to create or join existing groups and may belong to several groups simultaneously. Constraints can be placed on peer groups to implement security policies that control how and which peers may join.

Peer groups are designed to address several requirements, the first being security. The second is to provide an effective scoping mechanism that split the network into specialised domains, known as abstract regions which control the search space. Peers, within the network, share several peer group services which include the Discovery Service, Membership Service, Access Service, Pipe Service, Resolver Service and the Monitoring Service. The collective use of these peer group services provides the core functionality most P2P applications require.

The central idea behind JXTA is the concept of services, which are referred to as *modules*. JXTA supports two types of services called Peer Services and Peer Group services. Peer services are implemented and used by a single device. If the device is disconnected then the Peer services it provides are lost. Peer Group services are implemented on numerous peers and shared within the group. When a single peer in the group is disconnected you only lose the services provided by that device and devices are free to re-discover the same service provided by another device.

Services are abstractions, which can be used to hide the underlying implementation details regarding how the service is created. For example the implementation could be a Java class, or a jar file. At an abstract level services are described in a standard way and the implementation details are left to the device manufacturer. Each service is known as a network behaviour, which can be discovered and used by any other device within the group. JXTA services provide a flexible means of addressing interoperability between different implementations.

Pipes are one of the main mechanisms for sending messages between devices, which support both asynchronous and unidirectional communications. The message object can support any arbitrary data such as binary code or Java objects serialised as XML. Pipes are known as endpoints which may be input and output pipes mapped to network interfaces such as TCP/IP. This is dynamically performed at runtime.

XML advertisements are used to advertise networked resources such as peers, services and pipes. One of the key benefits of advertisements is that they are language neutral XML documents, which means that they can describe and advertise the existence of any resource irrespective of the programming language it was developed in or the underlying platform or transport protocols it uses.

Discovering resources is achieved by searching for advertisements. If a local advertisement is found then the device can use it otherwise JXTA searches for the advertisement remotely. Advertisements have a lifetime that specifies the availability of the resource. Using TTL values, resources can be deleted without having to use centralised control. Extending the lifetime of a resource can be achieved by republishing an advertisement before the previous advertisement expires.

Each resource within the JXTA network is identified using a globally unique ID, which is created using the JXTA J2SE binding. In the current JXTA specification there are six entities that use JXTA IDs. These are the *Peer*, *Peer Group*, *Pipe*, *Content*, *Module Class*, and the *Module Specification*. JXTA IDs are represented as Universal Resource Indicators (URIs) which are persistent location-independent identifiers.

IDs provide a level of abstraction that allow every network resource to be discovered and referenced in a standardised way without having to consider the underlying implementation details. This provides a unified addressing scheme that allows devices with different addressing schemes to interoperate. For example devices that use IEEE 802.15.4 (Zigbee) [IEEE Standards Association 2005] can communicate with devices that use 802.11x using the unified JXTA ID mapped to the underlying transport protocols being used – the conversion between standards is invisible to the device.

The JXTA specification has matured, and has considerable support from industry and academia alike. It is generating a great deal of interest within the ubiquitous and pervasive computing domains and research initiatives are currently assessing how it can be used in the digital home.

JXTA provides several discovery specifications, however they are somewhat restrictive because services are not discovered based on the capabilities individual devices or the services they provide support. The discovery process is based on pre-determined syntactic

descriptions. This technique is efficient when using pre-determined core framework services, however it becomes more problematic when discovering application specific services that are ad hoc in nature. These types of services are non-standard services that provide access to the devices underlying functions. As such these functions will be numerous. The current version of JXTA does not provide any mechanisms to discover services based on semantic descriptions that describe the behavioural aspects of the service. Additional core services need to be developed that extend the existing JXTA specification to address this requirement. This will enabled devices to automatically compose devices and services without any human intervention.

Other variants of P2P computing exist that are converging with home computing such as Instant Messaging (IM) [Shigeoka 2002], which has seen a significant growth in recent years. Instant Messaging follows a similar path as P2P in that the concepts have been around for some time. Mechanisms that allow one-to-one and group chatting have been around long before current IM solutions. Examples of such systems are Unix talk [Burk 1998] and Internet Relay Chat (IRC) systems [Douglas 2004], which are extensions of Unix talk. P2P is also being used to extend the gaming experience through distributed on-line game play. A technology generating a great deal of interest within this area is Jabber [Lee 2002].

2.5 The Semantic Web

The term 'Semantic Web' was coined by the inventor of the WWW, Tim Berners-Lee [Berners-Lee 2000]. Berners-Lee had a two stage view of the WWW. The first stage was to create a collaborative medium that allows authors to develop and host interconnected Web pages using HTML and the concept of hyperlinking. The second stage was to make the Web understandable in order to make data processable by machines as well as humans – this second stage will result in a 'Semantic Web'.

There is nothing mystical about the Semantic Web and people often frown upon the idea of making a machine intelligent and thus threatening. Berners-Lee clarifies the term by stating:

"A Semantic Web is not Artificial Intelligence. The concept of machine-understandable documents does not imply some magical artificial intelligence which allows machines to comprehend human mumblings. It only indicates a machine's ability to solve a well-defined problem performing well-defined operations on existing well-defined data. Instead of asking machines to understand people's language, it involves asking people to make the extra effort [Berners-Lee 1998]."

Daconta [Daconta 2003] makes reference to where the 'Smarts' in data resides. Traditionally data is propriety, which means it can only be accessed and understood by a purpose built

application. The data itself is not transitive and can only be accessed by pre-defined functions, exposed by the application – if you do not have the software, then you cannot access the data. Daconta states that in propriety data the 'Smarts' reside in the application and not in the data itself.

The introduction of XML has overcome this limitation and made information accessible within a single domain. The data itself resides outside the application and as a result the 'Smarts' reside within the data and not in the application. Doconta defines this type of data as application independent, which is smart enough to be transferred between applications within a single domain. The XML paradigm can be further extended to ensure that data is incorporated within and across multiple domains and is structured using taxonomies and classification hierarchies. The true power of taxonomies becomes evermore apparent when the data adopts the principles of ontology, and incorporates rules that enable information to be inferred from existing data using logical definitions. The word ontology derives from the Greek words 'onto' (being) and 'logia' (written or spoken discourse). There are many theories of ontology dating back to Aristotle ranging from 'concepts of being' to 'knowledge representation and information reuse.' A more detailed discussion on ontology is presented in Section 2.5.1 on page 46.

The Semantic Web is widely scoped and it is said that applications will be employed in various guises. The technologies surrounding the Semantic Web are not solely designed for the WWW, but rather define a set of tools and ontological languages that address the problem of semantic interoperability. These tools are becoming more widespread and are used within the areas of Sales Support, Strategic Vision, Marketing, Decision Support, Corporate Information Sharing and many more [Daconta 2003].

The fundamental issue the Semantic Web addresses is semantic interoperability. XML paved the way for syntactic interoperability, however it is important that this is extended to incorporate semantic interoperability to ensure that information is not just dumped in files and databases. The idea is to dress up this information and put the 'Smarts' in the data itself and enable syntactic and semantic interoperability within and across different domains.

Heflin [Heflin 2003] states:

"the goal driving the Semantic Web is to automate Web-document processing. To that end, researchers are developing languages and software that adds explicit semantics to XML's content structuring aspects. A Semantic Web language lets users create ontologies that specify standard terms and machine-readable definitions. Information resources (such as Web pages and databases) then commit to one or more ontologies, thus specifying which sets of definitions are applicable to a specific resource. For example, an ontology about animals

might explicitly state the class 'Dog' is a subclass of 'Mammal' and that the classes 'Mammal' and 'Fish' are disjoint. Logical reasoning systems can use these statements to deduce additional information that was not explicitly stated about the terms in the resource."

He further highlights the main challenges facing the Semantic Web:

"although a standardised Web ontology language will be a major step forward, several challenges need to be addressed before the Semantic Web can become a 'Pragmatic Web' – an online environment that not only helps computer systems find information, but also helps ordinary people accomplish tasks and get practical work done. The challenges include:

- *Getting information into the appropriate format*
- *Scaling Semantic Web technology to handle 'Web size' data*
- *Creating, maintaining and integrating ontologies*
- *Using the Semantic Web to describe and compose Web Services*
- *Handling inconsistent data and*
- *Determining what to trust."*

Heflin highlights some interesting challenges, more notably the idea that we need to get information into an appropriate format as well as creating, maintaining and integrating ontologies; and determining what to trust. He describes a "chicken and egg" problem whereby if semantic web content was available then more systems and agents would use the Semantic Web for search tasks and if it were used in more searches, more content providers would be willing to provide information in the specified format. This is an interesting challenge.

2.5.1 Ontology

A brief definition of ontology was presented above and we highlighted that many theories have been presented ranging from 'concepts of being' to 'knowledge representation and information reuse.' Decker *et al* [Decker 2000] define ontologies as:

"a shared formal conceptualisation of a particular domain which provides a common understanding of topics that can be communicated between people and application systems."

Whilst Gruber [Gruber 1993] states *"an ontology is an explicit specification of a conceptualisation."*

Following a similar description Uschold and Gruninger [Uschold 1996] define an ontology as:

"a shared understanding of some subject area which helps people or processes achieve better communication, interoperability and effective reuse. The Ontology embodies a conceptualisation – definitions of entities, their attributes and relationships that exist in some domain of interest. The conceptualisation is explicitly represented."

In the technical view of ontological engineering, ontology is the vocabulary for expressing the entities and relationships of a conceptual model for a general or particular domain, along with semantic constraints on the model that limits what the model means. Both the vocabulary and the semantic constraints are necessary in order to correlate that information model with the real-world domain it represents. Complex ontologies far exceed the capabilities of simple ontologies such as taxonomies and catalogues, in that they are capable of consistency checking, providing completion, interoperability support, validation and verification, comparative and customised search, and exploiting generalisation and specialisation information. The following sections explain this distinction by defining weak and strong ontology representations.

2.5.1.1 Weakly Defined Ontology

Taxonomy is based on classification, which ensures that things are organised into logical hierarchies. The hierarchy itself is represented as an upside down tree. Branches within the tree are defined as nodes, with the top node being the most general. As nodes move further down the tree, they become more specialised. For example, a 'Dog' is a more specialised concept than an 'Animal' concept therefore the node 'Dog' will appear under the node 'Animal'. The links between nodes are referred to as subclassification and superclassification. For example the node 'Dog' appears as a subclassification of 'Animal' whilst 'Animal' appears as a superclassification of 'Dog'.

Taxonomies have proved to be a powerful tool for classifying information semantically (in terms of taxonomies, this is usually defined as weak semantics or meta-data). By definition this means that they are directly associated with technologies that focus on knowledge representation such as thesauri, conceptual models and ontologies. Taxonomies are often referred to as semantically weak representations because of their inability to express information using rich modelling primitives. At the very best taxonomies can only provide a simple model capable of making simple distinctions between objects, which primarily focus on browsing and navigating information structures.

There is a subtle distinction between semantically weak representations and semantically strong representations. Something is a subclassification or a superclassification of an object within a taxonomy, however semantically strong representations enable us to define nodes using richer model constructs such as *disjointTo, equivalentTo* as well as using properties to describe the individual characteristics a class supports. Subclassification and superclassification make the taxonomy structures ill-defined and semantically weaker than other structures such as conceptual models and ontology. McGuinness [McGuinness 2001] clarifies this point by stating:

"In these organisation schemes, it is typically the case that an instance of a more specific class is also an instance of the more general class but that is not enforced 100% of the time. For example, the general category Apparel includes a subcategory Woman (which should more accurately be titled Women's Apparel) which then includes subcategories Accessories and Dresses. While it is the case that every instance of a Dress is an instance of Apparel (and probably an instance of Women's dress), it is not the case that a Dress is a woman and it is also not the case that a Fragrance (an instance of a Women's accessory) is an instance of Apparel." She further states "Without true subclass (or true "isa") relationships, we will see that certain kinds of deductive uses of ontologies become pragmatic."

The Thesaurus is probably one of the most common classes of taxonomy. It is classed as a semantically weak classification that enables information to be structured and ordered in a known way so that equivalence, homographic, hierarchical and associative relationships among terms can be displayed clearly and identified by standardised indicators. A thesaurus is primarily used to aid information retrieval based on the rough associations between any terms. This ensures that concepts are described in a consistent way and provides a tool for users which enables them to drill down until required information is found.

McGuinness [McGuinness 2001] classifies thesauri as simple ontologies and states:

"thesauri are controlled vocabularies. These types of ontologies prove useful and common term usage provides a starting point for interoperability. They are used for Web site organisation and navigational support. In this sense they are a generalised hierarchy of terms which can be further exposed to reveal relevant subcategories. Using hierarchical tree structures provides the user with a realistic expectation of the site and enables the user to quickly determine if the site contains the information they are looking for. This type of functionality can be viewed as a browsing tool, which tags content to aid browsing and searching."

2.5.1.2 Strongly Defined Ontology

Conceptual models extend the capabilities of taxonomies by modelling a particular domain to form a complex knowledge representation. The domain consists of entities, which have relationships with other entities, and possess attributes with associated values. Conceptual models extend the capabilities of typical taxonomies by fully implementing the ability to capture the subclass relationships between parent and child classes. These models use the object-oriented paradigm to construct complex knowledge domains which consist of the meta-level and the object-model level. The meta-level defines the classes, the relationships and the properties, whereas the object-model level defines content models. Ontologies can be seen as conceptual models and more specifically logical models. Logical models are defined

as the combination of axioms, inference rules, and theorems. Axioms and inference rules are used to prove theorems about the domain represented by a particular ontology.

The Resource Description Framework (RDF) [Decker 2000, W3C 2005] is a simple model and is based on XML syntax, which has been a W3C recommendation since February 1999. Its primary function is to describe resources using URLs. RDF differs from XML in several different ways. The main difference relates to how the formats apply meta-data. XML applies meta-data to the internal structures of an XML document whilst and RDF document focuses on providing meta-data about the external information associated with a document such as 'Author' and the date the document was created.

The RDF Model is based on a collection of triples. A triple is the name given to RDF statements, which contain three parts – the subject, predicate and object. The subject is a resource, which can be either an electronic source such as a Web page or it can be a concept like 'Car'. A resource can be identified as anything that can be given an identity [Daconta 2003]. The predicate is a verb that links the subject and the object together. For example the predicate in the following sentence "John throws the stick" is throws, which links the subject (John) to the object (Stick). The object is a value associated with the subject via the predicate. The object may be another subject or resource or it may be a literal value such as a string or a numerical value.

The RDF structure itself can be represented using three different formats; RDF/XML, a triple notation called N3 or a graph-triple notation. In addition to the simple triple model, RDF contains two further features which deal with collections, more formally known as a Bag object and Reification. The Bag feature is self-explanatory and allows groups of resources or values to be combined. Reification is rarely used and focuses on high-level statements used to describe other statements.

Although RDF offers distinct advantages over raw XML, it has not been widely accepted and its uptake has been slow. This can be attributed to three reasons. It is difficult to embed RDF within XML and as a result it is not easy to validate it. A second reason is that parts of RDF are complex which make the development process significantly more difficult than XML development. RDF allows metaphors to be mixed, which means that RDF documents are capable of using terms from difficult representations provided by different organisations such as linguistics, object-oriented concepts and relational data [Daconta 2003]. This is advantageous in one sense because it provides for a more integrated environment that promotes knowledge sharing; however this also causes a great deal of confusion.

The third problem can be directly related to the hierarchical constructs of RDF. It proves difficult for document authors to arrange triples into a hierarchical structure using RDF in its

raw form. The document soon becomes unwieldy and difficult to follow or maintain – however tools do exist, that abstract the complexities away from the user.

RDF provides a model capable of linking resources together in a directed graph format; however it is too simplistic to capture the true semantics of information. RDF Schema (RDFS) was developed to extend RDF and enable information to be represented as classes and properties of classes with associated values. This allows class definitions to be represented as inheritance hierarchies. RDFS can also further constrain the model by placing domain and range restrictions on properties [Fensel 2003]. RDFS, like RDF is a simple model, which provides a set of simple standardised resources and properties that enable authors to create ontology-based vocabularies, and is based on an object-oriented paradigm. Whereas RDF describes information at the instance level, RDFS extends this to represent information at the class level. It allows the author to model information using object-oriented principles, which is restricted to the development of classes and data that captures object behaviours – RDFS is concerned with modelling data not behaviours and enhances the modelling capabilities of RDF or XML not only to include classes, and properties of classes but to also define complex relationships between classes and properties, such as *subClassOf* and *subPropertyOf*, making classes and properties transitive.

Another specification for describing data is that of Topic Maps (TM) [Le Grand 2001]. Topic Maps are defined as a context-oriented index which sits above a set of documents. This indexed-based overlay enables content based navigation over resources, which acts like a taxonomy that describes, classifies and indexes a desired information space. TMs are not new and appeared before XML. They were based on the Standard Generalised Markup Language (SGML) [Goldfarb 2002] representation and became an ISO Standard (13250), which today has two interchangeable syntaxes – XML and SGML. The more common representation is XML and current TMs are usually referred to as XML Topic Maps (XTM). The key concepts surrounding TMs are topic, association, occurrence, subject descriptor and scope. A topic is defined as anything that can be a distinct subject of interest – the topic itself usually acts as a proxy for a particular subject. Capturing subjects within a TM enables us to make assertions about the subject. An occurrence is defined as a resource that provides us with some information about a topic. The occurrence is described using a URI and has an associated data value, which can be of different types, however unlike RDF, the value may not be another resource. This is one of the fundamental limitations of TM and where RDF provides a more complex form of linking. An association is defined as a relation between one or more topics. A subject descriptor is defined as something that can be a resource, which has an associated information representation called a topic. The scope is defined as the context of the topic, its

occurrences, resources and associations. The concept of scope is the same as a namespace used in current markup languages.

In formal languages there is a vocabulary, which can be defined as a language that has a syntax and associated semantics for the objects of that syntax. The primary function of ontologies is to reduce the models of interpretation of specified vocabularies in order to remove as much ambiguity as possible. No other model type, for example taxonomies, does this. Consequently these models rely on the human to understand the semantics and resolve any ambiguities that may exist. The view is that machines should be responsible for this level of processing so the reliance on human intervention can be minimised.

2.5.1.3 Ontology Specifications

Many ontology specifications have been developed over the last twenty or thirty years. The Simple HTML Ontology Extensions (SHOE) [Heflin 1998] specification was one of the first languages that used ontologies for direct use on the World Wide Web (WWW) and was viewed as the blueprint for the Semantic Web [Berners-Lee 2001]. SHOE combines the features of mark-up languages and borrows the characteristics from both predicate logic and frame-based systems. SHOE is designed directly in HTML and XML documents, however it provides more benefits if it is embedded in XML because the extensive tools available, such as the Document Object Model (DOM) [Goldfarb 2002], which can be used to perform validation at the XML level. The SHOE syntax however still has to be parsed by SHOE-aware software.

SHOE attempts to enhance interoperability between distributed Internet agents, by using shared ontologies, prefix naming, prevention of contradictions, and locality of inference rules [Fensel 2003]. Before SHOE can be used, an ontology needs to be located in a centralised repository, which may consist of a number of Web pages that categorise ontologies, or the repository itself may be more complex and enable the ontology to be annotated with meta-data indicating key characteristics. This is said to provide a better search mechanism, however if no ontology exists then a new ontology needs to be constructed from scratch.

In SHOE the Web page or the XML document is annotated. This means that SHOE-based tags are inserted into the document. These documents are published on the Web and discovered and used by SHOE-based proprietary software capable of understanding the SHOE language. Documents are harvested using the SHOE Web Crawler called Expose, which searches Web pages with SHOE syntax and stores these documents in the knowledge base. The documents themselves can be used and the SHOE syntax extracted and processed using a reasoner such as RACER [Haarslev 2001].

The XML Ontology Language (XOL) [Karp 2005] is a language for ontology exchange, inspired by Ontolingua [Farquhar 1997] and the Ontology Markup Language (OML) specification [Kent 2005]. Ontolingua defines ontologies using the LISP programming language and OML uses conceptual graphs. The initial XOL specification is based on a Document Type Definition (DTD) schema [W3C 2005], however this was updated to the XML Schema specification [W3C 2005] by Dimitrov [Dimitrov 2000]. The main difference between XOL and its predecessors is its use of data definition syntax. Other research initiatives such as the Darpa Agent markup Language (DAML) [DAML 2003a], DAML-ONT, MCF, OntoBroker, On-To-Knowledge and OIL [Fensel 2001], where also developed in an attempt to create a de facto ontology standard.

OIL is a Web-based language and inference layer for ontologies, which combines primitives from frame-based languages with the formal semantics and reasoning services provided by description logics. OIL was the first ontology language to fully incorporate standards from the W3C (RDF/RDFS as well as XML and XML-Schema). However, OIL extends RDFS by adding additional language primitives not present in the RDFS specification. OIL marked a significant advance and boosted superior capabilities not evident in languages such as CycL [CyCorp 2002], KIF [Genesereth 1991], Ontolingua [Farquhar 1997] or any of the ontology languages described above. It unifies three important aspects provided by different communities; epistemological modelling primitives as provided by the frame community, formal semantics and efficient reasoning support as provided by description logics, and a standard proposal for syntactical exchange notations as provided by the Web community.

Instead of continuing with different languages for the Semantic Web a group of researchers created the joint US/EU ad hoc Agent Markup Language Committee to create a new ontology language called the Darpa Agent Markup Language + Ontology Inference Layer (DAML+OIL) [DAML 2003a], built on both OIL and DAML-ONT. DAML+OIL constituted the most semantically expressive language available for WWW documents.

The DAML+OIL specification was submitted to the W3C, which became the basis for the Web Ontology Language (OWL) [Smith 2005] specification, which to date is considered the *de-facto* specification for describing ontologies on the Web.

2.5.1.4 Consensus Ontologies

Stephens *el al.* [Stephens 2001] describe the problems associated with information retrieval and illustrate that although some sophisticated techniques exist that use ontologies, to date there is no comprehensive ontology that can solve the problems associated with information retrieval. Even if you could create such an ontology it would be so eclectic that no one would adhere to it. Web developers could use a common terminology with agreed semantics,

however this solution is highly improbable. Web developers could use there own terminology and explicitly provide translations to a global ontology, however this is difficult and as a result highly unlikely.

A possible solution provided by Stephens *et al.* describes how Web developers could use small, localised ontologies related indirectly with the assistance of agents. The solution is based on the multiplicity of ontology fragments, representing the semantics of the independent sources that can be related to each other automatically without using a global ontology. Direct relationships between a pair of ontologies can be determined indirectly using a semantic bridge. The resultant merged ontologies provide a semantic characterisation of the set of sources and their domains, and effectively create a single large ontology to serve as a global hub for interactions.

Stephens *et al.* further argue that a consensus ontology is perhaps the most useful for information retrieval by humans because it represents the way most people view the world and its information. He makes the following statement:

"If most people wrongly believe that crocodiles are a kind of mammal, for example, then most users would find it easier to locate information about crocodiles located in a mammals grouping, rather than in reptiles where it factually belongs."

The precision and recall of information retrieval measures are based on some degree of match between a request and a response. The length of a semantic bridge between two concepts can provide an alternative measure of conceptual distance and an improved notion of information relevance. Previous measures relied on the number of properties shared by, or the number of links separating two concepts within the same ontology. These measures not only require a common ontology, but also fail to account for the density or paucity of information about a concept.

Although this is an interesting approach it is not clear how easy it is to develop agents to perform mappings to create semantic bridges. Ontologies will be serialised using different specifications so interoperability between different serialisations is paramount. It is not clear how Stephens *et al.* propose to address this problem. An assumption needs to be made regarding the serialisation whereby the representation is standardised, however the concepts themselves remain totally unconstrained. Extending this further it is difficult to determine how effective there algorithms are in terms of performing mappings using rich complex ontological constructs such as those evident in the OWL specification. Typically ontology engineers use real-world knowledge to create, merge or align ontology fragments, which takes considerable effort. Trying to automate this process is not easy. However they do argue this point above based on precision and recall.

Furthermore it is not clear how computationally expensive there approach is or how easy it is to maintain the process. Localised ontologies will be subject to continual change, consequently agents will need to maintain every semantic bridge it is responsible for. This is somewhat simplified because there will be numerous agents which are only concerned with a small proportion of the global ontology. This is analogous to the concepts used in P2P computing whereby routing tables are managed for neighbouring peers only. Consequently these systems are scalable; however they are computationally expensive in ad hoc environments where continual change is the norm. Maintaining a global view may be easier within controlled environments such as organisational LANS, however maintaining a global view in ad hoc environments is more costly.

2.5.1.5 Ontology Evolution

It is generally agreed that describing information using ontologies provides a better solution to discovery than attribute-value pair matching. Ontologies provide a semantic bridge between different concepts providing mechanisms that help systems to proactively understand and learn the relationships between different terminologies. This allows systems to communicate with each other and understand terms that are syntactically different but semantically equivalent.

Using ontologies for semantic interoperability proves successful in controlled environments, however there are a number of challenges that need to be addressed such as semantic interoperability, ontology heterogeneity, ontology merging and alignment and global agreement on what constitutes a concept including how it should be described. An approach used by Aberer *et al.*, [Aberer 2003] is to capture knowledge through gossiping. Their approach aims to interconnect peers within a P2P network via user-defined schemas to share and incrementally evolve the search capabilities within the network. Their approach assumes a large amount of data exists and that they have been organised and annotated according to local schemas, which is not always the case in distributed networks. This technique primarily focuses on creating mappings between ontologies based on the similarities found between terms and relationships. This process requires an experienced knowledge engineer to have an understanding of all the ontologies being mapped which must be continually maintained as and when concepts are disproved, links are broken or new links added.

Noy *et al.* [Noy 2000] describe an algorithm they have developed called PROMPT that provides a semi-automatic approach to ontology merging and alignment. It performs some tasks automatically and guides the user through other tasks by taking two simple ontologies as input and attempting to merge them into a single ontology. The algorithm merges the ontologies based on similarities between classes, slots and bindings between slots. This

presents an interesting solution, however the merging process is based on the subjective opinions of the user merging the two ontologies and suggests the person merging the ontologies is an experienced knowledge engineer capable of creating the correct mappings.

The same problems are experienced in the Chimaera system developed by McGuinness *et al.* [McGuinness 2000], which provides assistance with the task of merging knowledge bases (KBs) produced by multiple authors. This is a web-based ontology editor that merges two or more ontologies together based on identical terms and subsumption relationships between terms. Again this approach experiences the same short-comings as PROMPT in that an assumption is made that experienced knowledge engineers will carry out the merging process.

ONION [Mitra 2000] combines two separate ontologies to form an articulated ontology. Rather than merging, ONION performs an alignment between two ontologies by capturing the semantic gap between the two. This approach is similar to Aberer's approach in that the technique acts like a mapping between two different representations. The process of creating the semantic gap involves semantically relating classes and creating and managing semantic bridges. ONION uses a semi-automatic approach, which relieves the user from having to maintain the bridges; however this approach assumes that a domain expert, knowledgeable of both structures, creates the semantic bridges.

All these approaches require human intervention during the ontology construction phase and although there are semi-automated tools that aid the knowledge worker there are no mechanisms to completely automate this process. The challenge is to allow knowledge to be distributed and discovered using advances made in global communications and distributed systems technology, which enable ontologies to be evolved based on general consensus without any human intervention.

2.5.2 Semantic Web Services

McIlraith et al. [DAML 2003b] highlight that there is a need to describe Web Services in terms of their capabilities in an unambiguous, computer-interpretable language. Advances made in Web Service technologies and research carried out by the Semantic Web community could provide a means to achieve this vision by combining these technologies together to produce Semantic Web Services. McIlraith et al. describe how the DAML for services (DAML-S) upper service ontologies can be used to describe Web services in terms of their capabilities. They illustrate how DAML-S builds on the complementary technologies used by Web Services such as WSDL and SOAP to enable dynamic service discovery, composition and execution. McIlraith *et al.* provide a clear justification for using DAML-S to describe the capabilities of services using machine-processable semantics, which WSDL alone is incapable of doing.

Paolucci et al. [Paolucci 2002a, Paolucci 2003] describe a matching engine that determines the similarity between a service request and a service description by evaluating the inputs and outputs they define. They use a term called "sufficiently similar"; in its strongest sense a service description and a service request are sufficiently similar when they describe exactly the same service. They state that this is too restrictive, because advertisers and requesters have no prior agreements on how a service is presented. A restrictive criterion on matching is bound to fail to recognise similarities between service descriptions and service requests. To accommodate a softer definition of "sufficiently similar" Paolucci *et al.* explain that there is a need to allow matching engines to perform flexible matches based on the degree of similarity between the service request and the service description.

One of the main problems with the work carried out by Paolucci *et al.* is that it only performs matches using the service profile. It does not process the remaining service ontologies to determine if the information provided in the service request can be directly mapped onto bindings described in the WSDL file associated with a particular service. Their research clearly indicates that semantic searches provide a better alternative to attribute-value pair matching, however they provide no mechanisms for automated service composition.

Maedche *et al.* [Maedche 2003] provide an assessment of service-driven systems and describe the need to converge three separate technologies – Web Services, P2P technologies and the Semantic Web. They argue that combining these technologies will allow services to be identified, located and invoked. This new paradigm is important to the development of service-enabled systems, however this is no easy task and the integration process itself gives rise to new complexities such as locating and integrating services on the fly, semantic interoperability, data heterogeneity and process mediation. Maedche *et al.* make a strong case for combining several active areas of research and explain the importance and difficulties with the integration process itself.

In this thesis we describe how our work is heavily reliant on distributed services within P2P networks and illustrate how we aim to capitalise on the efforts made within the research disciplines discussed in this chapter to better describe, discover and automatically compose networked appliances based on semantic descriptions that describe the capabilities of service requests and service descriptions.

2.6 Summary

There are many solutions that allow devices to be interconnected within the home environment, however little advance has been made to abstract the complexity away from this process. Technology is becoming more pervasive, consequently trying to manage solutions and their associated configurations is becoming more difficult. Several research initiatives in

the area of communications and service-oriented architectures promise to provide solutions that realise a seamless integration between heterogeneous devices, however to date few solutions have produced any convincing results.

Frameworks such as OSGi, UPnP, DLNA and HAVi, including new projects such as VHN, MediaNet, RUNES, ePerSpace, VisNet and Future Home are attempting to integrate devices, however they are are managed and controlled via centralised providers. Services are discovered and composed based on proprietary communication and middleware protocols. Interoperability issues are addressed using agreed standards and although this is not impossible it is not clear whether a single standard is capable of addressing all issues. The goal must be to utilise existing open standards as much as possible and interoperability mechanisms must be developed that abstract the underlying implementation details allowing any standard to be used and seamlessly integrated.

Furthermore the solutions described in this chapter do not provide any mechanisms to enable devices to automatically discover and compose devices and services. Compositions are carefully choreographed and control is based on application specific serialisations. Some solutions require separate hardware adapters to convert appliances into networked appliances. This is somewhat restrictive since distributed computing and service models are becoming increasingly more pervasive. As such devices and services are becoming more heterogeneous in nature. Consequently managing such a framework will be more complex where the amount of control placed on device and service integration becomes more difficult. Different device and service providers will use different communication, middleware and service standards. As such interoperability is a problem that will require more effective solutions. As such new architectures need to be developed.

The P2P networking model is seen as a key enabling technology that will extend the reach of devices connected to each other via global communications. As well as sharing digital content, devices will be able to share and discover network behaviours provided by other devices connected to the network. This in effect enables devices to share hardware resources and services. Like home networking middleware solutions, P2P also supports several techniques that have both strengths and weaknesses. Early P2P implementations such as Napster proved successful, however these early solutions suffered from a number of limitations, which include single content type sharing; and a reliance on client-server technology. The primary difficulty with solutions such as Napster is it’s central point of control – switching off the Napster servers in effect disables the search mechanism and as such content cannot be shared or discovered.

DHT-based P2P implementations adopted a more decentralised model. Unlike Napster, these new P2P models are more difficult to control because no central server is used. However these solutions are not without their own problems. Maintaining a consistent distributed index in DHT-based solutions is expensive because most time is spent updating indices. DHT-based solutions provide an efficient mechanism for data access, however costs are exponential as the number of peers that continually connect and disconnect increases. If a DHT approach is not used then computational costs are reduced however an exhaustive traversal of the network is required, which results in network flooding. This said these solutions work well in structured networks whereby control can be placed over the network topology. For example an organisational network could be controlled to ensure that the frequency in which nodes join and leave the network is kept to a minimum ensuring that DHT table updates are negligible. However these solutions are not as effective in unstructured networks, such as global P2P networks, whereby devices will continually come and go. Environments that are highly ad hoc and mobile in nature are subject to continued change resulting in node DHT table update algorithms continually managing all changes that occur.

All the above mentioned P2P models primarily focus is on sharing digital multimedia files such as MP3 and AVI. None of these solutions provide any mechanisms to publish and discover services. As distributed computing models move towards service-oriented architectures, it is becoming more important for P2P implementations to support service technologies. A new set of specifications called JXTA has realised this and is a new breed of P2P that supports the discovery of both digital content and services. This marks a significant advance in P2P technology. It is not sufficient to just host services but to also effectively discover and use them. Services that are deterministic as is the case with JXTA core services are easy to discover and use, however custom services are more problematic. We envisage that there will be a large number of different services. Consequently it is impossible to predetermine all the interfaces these services provide.

The current JXTA specifications allow custom services to be hosted, however the discovery specifications provided by JXTA are somewhat restrictive because the discovery process is based on predetermined syntactic descriptions. This technique is efficient when using pre-determined core framework services, however it becomes more problematic when discovering application specific services that are ad hoc in nature. The current version of JXTA does not provide any mechanisms to overcome these limitations. Additional services are needed that extend the existing JXTA specification to provide better service discovery mechanisms.

Services and the requests for services themselves need to utilise advances made within the Semantic Web and Semantic Web Service communities. Alternative mechanisms are required that overcome the inherent limitations associated with simple syntactic matching such as

attribute-value pair matching. This will allow devices to discover and use services based on rich ontological descriptions that describe the behaviours of services, thus providing better matching mechanisms for service discovery.

Several approaches within the Semantic Web, ontology engineering and Semantic Web Services communities are trying to address this issue. We began by arguing that although the thesaurus is probably one of the most common classes of taxonomy, it is classed as a semantically weak classification that only enables information to be structured and ordered in a known way so as to aid information retrieval based on the rough associations between terms. Although thesauri have proved useful they are somewhat restrictive because they use limited modelling primitives to describe concepts, the properties they support and the relationships they have with other concepts.

Another serialisation is RDFS and standards built on top of RDFS, include TM, XTM, DAML+OIL and OWL. All these are classed as ontology languages with distinguishing features being in their ability to describe concepts. OWL is the most descriptive ontology specification allowing complex knowledge structures to be modelled. OWL is designed to reduce the models of interpretation within different domains, which aims to remove as much ambiguity as possible making it easier for information to be processed by machines and humans alike.

Serialising ontologies is a manual process. Research carried out by Stephens *et al.*, suggest that this is restrictive and it would be better if this process could be automated using agents. There approach is interesting and will become increasingly more important. However it is not clear how easy it is to develop agents capable creating and managing semantic data. Ontologies will be serialised using different specifications so interoperability between different serialisations is paramount. It is not clear how Stephens *et al.* propose to address this problem. It is difficult to determine how effective there algorithms are in terms of performing mappings using rich complex ontological constructs such as those evident in the OWL specification. Typically ontology engineers use real-world knowledge to create, and merge or align ontology fragments, which takes considerable effort.

Aberer *et al.* use an approach that assumes a large amount of data already exists and that they have been organised and annotated according to local schemas. This process requires an experienced knowledge engineer to have an understanding of all the ontologies being mapped which must be continually maintained as and when concepts are disproved, links are broken or new links added. This is costly and somewhat problematic because the knowledge engineer is seen as the bottleneck; his opinions are subjective and he is susceptible to human fallibility.

Noy *et al.*, Chimaera and ONION also propose similar approaches; consequently the same limitations are apparent.

Several researchers are developing semantic service solutions, which use OWL and DAML+OIL serialisations, however the matching process is limited and does not support automatic discovery and composition of ad hoc services within highly disruptive network configurations. Paolucci *et al.* have developed a semantic matching algorithm; however it only performs matches using an abstract service profile as provided by the OWL-S specification. This is adequate for service discovery; however it does not aid dynamic service composition. It does not process the remaining service ontologies to determine if the information provided in the service request can be directly mapped onto signatures described in the service interface.

There are several other industry lead initiatives such as the Business Process Execution Language for Web Services (BPEL4WS) [Andrews 2005], the Component Service Model with Semantics (CoSMoS) [Fujii 2004], the Anamika [Chakraborty 2003] framework, and the Integrated Service Planning and Execution (ISP&E) [Madhusudan 2004] framework, which provide standards to compose Web Services in controlled environments. The major limitation with these standards, however, can be directly attributed to the inability to compose services on-demand where the location of services are not known or if they exist [Sirin 2003]. The plethora of mobile devices is on the increase and the number of services they expose will be numerous, therefore it is paramount that we develop mechanisms that discover, compose and execute services on demand, without having to carefully choreograph the composition and execution process beforehand.

It is paramount that ontologies are used to better describe what services require and what services provide if we are to develop frameworks capable of automatically discovering and composing devices within ad hoc environments. Services need to be described semantically to describe their capabilities in an unambiguous machine-interpretable language that allows networked appliances to automatically form compositions with each other based on the available functions within the environment at any given moment. This will allow devices to manage the integration process and self-adapt to environmental changes as and when they occur, whilst minimising the amount of disruption.

2.6.1 Challenges

This Chapter has described the key research within the areas of Networked Appliances, Home Networking, P2P technologies and the Semantic Web. 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.

1. Interoperability mechanisms need to be defined that allow any device to be seamlessly integrated. Different device and service providers will use different communication, middleware and service standards. As such interoperability is a problem that requires a more effective solution.
2. A global view is paramount whereby devices and services can be discovered and integrated into new and existing configurations irrespective of where they reside within the global Internet. The challenge is to disperse the operational capacity of devices within the network by utilising P2P technologies so that functions can redundantly coexist and be discovered with local and global scope in mind.
3. Services and the request for services need to utilise advances made within the Semantic Web and Semantic Web Service communities. The challenge is to develop mechanisms that overcome the inherent limitations associated with simple syntactic matching such as attribute-value pair matching, to allow devices and services to be more accurately discovered and composed.
4. It is paramount that we develop mechanisms that discover, compose and execute services on-demand, without having to carefully choreograph the composition and execution process beforehand.
5. We can extend challenge three to define mechanisms that allow knowledge to be distributed, discovered and evolved based on general consensus without any human intervention. This will help support interoperability and ensure services and devices are more accurately matched.
6. We can also extend challenge four to allow devices to manage the integration process and self-adapt to environmental changes as and when they occur, whilst at the same time minimising the amount of disruption.

Chapter 3

3 Networked Appliance Service Utilisation Framework

3.1 Introduction

In Chapter 2 we argued that current networked appliance and home networking platforms are restrictive because they are heavily reliant on human intervention and carefully choreographed vocabularies. Such approaches lack flexibility and do not scale in ad hoc environments where little control can be placed on devices within the network or the services they provide. They do not provide any mechanisms to effectively disperse services within the network or discover those services using high-level semantics. The configuration process itself is inherently human centric and there are no mechanisms to allow the configuration and management of device configurations to be automated with little or no human intervention.

In this chapter we present our design for a Networked Appliance Service Utilisation Framework. This framework addresses the challenges discussed in Chapter 1 on page 1, which include service-oriented networking; service discovery; device capability matching; dynamic service composition, self-adaptation; and ubiquitous computing. The framework allows operational functions provided by devices to be dispersed within the home network; devices can interconnect with other devices over time to form high-level compositions; and devices can resolve terminology differences between vocabularies used to describe service interfaces and service requests. We begin this chapter by proving an overview of our framework.

3.2 Framework Overview

Our Networked Appliance Service Utilisation Framework (NASUF) is a Service-Oriented Middleware (SOM), which allows ad hoc services [Fergus 2003a] offered by service-enabled networked appliances [Mingkhwan 2004] to be dynamically discovered and composed within a P2P network devoid of any centralisation. Each device implements the core and secondary services that comprise NASUF as well as application specific services that disperse the functions devices provide as independent services within the network. For example a TV could have three application specific peer services; a visual service; an audio service; and a terrestrial TV receiver service.

NASUF services allow devices to be connected to the network to form relationships with other devices and self-adapt when environmental changes are detected. These services are the Distributed Semantic Unstructured Services (DiSUS) Manager, the Device Capability (DeCap) service, the Distributed Emergent Semantics (DistrES) service and the Semantic Interoperability and Signature Matching (SISM) service. Our framework is illustrated in Figure 3.1 and each service is described in turn below.

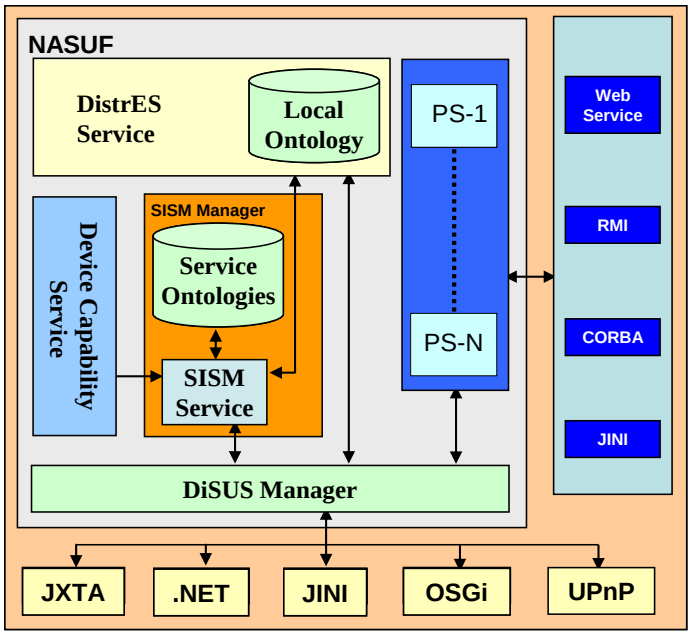

Figure 3.1 NASUF Framework

- The Semantic Interoperability and Signature Matching (SISM) Service performs dynamic service compositions between networked appliances in the P2P network based on device and service capability matching [Fergus 2005a]. This service is used to semantically match service requests with service descriptions. Any ambiguities found are resolved using the DistrES service, which is described below.
- The Distributed Emergent Semantics (DistrES) Service [Fergus 2003b] allows ontological structures, used to describe devices and the services they provide, to be evolved within the network based on general consensus. One of the key requirements within our research is to address the inherent terminology differences that will exist between different vocabularies used by different device manufacturers to describe the services their devices provide. The DistrES Service achieves this by evolving the knowledge structures provided by devices to create explicit mappings between terms that are syntactically distinct but semantically equivalent.
- The Device Capability (DeCap) Service [Fergus 2005a] determines the quality of the capabilities provided by devices, which include the hardware, software and network capabilities needed to execute services. The DeCap service is designed to ensure that

an overall quality-of-service (QoS) for a particular service composition is equal to or greater than the capability requirements defined within the service request.

- Devices will support zero or more application specific peer services (PS), designed to publish the functionality they provide as independent services. Peer services provide a level of abstraction that may map onto any service technology used, thus enabling service interoperability. Devices will discover and form compositions with services that reside locally on the device or remotely within the network to produce value-added services that yield functions that could not be performed by one single service alone.
- The DiSUS Manager [Fergus 2003a] is a core component that is implemented on every device. It manages all services and provides several interfaces that allow the device to be connected to any Service-Oriented Middleware (SOM) implementation, irrespective of the underlying network protocol. It provides mechanisms for device-to-device messaging, service discovery and mechanisms that allow devices to self-adapt based on environmental changes.

The DiSUS Manager is the core service each networked appliance must implement. This is a minimum requirement designed to enable networked appliances, independent of the capabilities they support, to effectively interact within the NASUF network. The remaining secondary services (DistrES, DeCap and SISM) must be either implemented locally on the device itself or discovered remotely within the network and bound to before the device is rendered fully functional. A device may implement some secondary services and outsource the remaining secondary service functionality to other more capable devices within the network. This feature provides a level of flexibility that allows interconnection between devices that support varied capabilities. For example a mobile phone may only implement the DiSUS Manager because the memory and processing constraints typically associated with this type of device and discover the remaining services within the network.

In the following subsections we present our framework design before concluding this chapter. The remaining secondary services provided by our framework are presented in Chapter 4

3.3 Distributed Semantic Unstructured Services (DiSUS)

One of the key requirements within NASUF is to enable devices to automatically connect to the network without having to register themselves or the services they provide with any third-party authority. When a device is switched on it must dynamically integrate itself and publish the services it provides. In addition, at any point, it must be free to disconnect and withdraw its services from the network. This section describes how this is achieved using a protocol developed within this research called Distributed Semantic Unstructured Services (DiSUS)

[Fergus 2003a]. This protocol implements mechanisms to distribute services within a P2P network and contributes additional knowledge to this area by enabling peers to semantically discover them dynamically based on device capability matching.

3.3.1 The DiSUS Protocol Requirements

Within this work, one of the challenges is to allow devices to exist within ad hoc networks and effectively publish the functions they provide as independent services. In order to achieve this it is paramount that the protocol addresses a number of key requirements, namely:

- ***Interoperability***: The protocol must support interoperability between different service technologies, middleware architectures and underlying protocols.
- ***Decentralisation***: Devices and the services they provide must be decentralised; every device that joins the P2P network must be capable of reaching any other device or service without using any centralised third party registry.
- ***Structured and Unstructured services***: Services must be described and discovered using pre-determined and non-determined vocabularies and interfaces.
- ***Dynamic environments***: Devices and the services they provide must be able to work in dynamically changing environments [Wilson 2002]. The base assumption is that devices and services will come and go over time.
- ***Intelligent Discovery***: Services must be described and discovered using semantic descriptions and processed using toolsets that have inferential capabilities [McIlraith 2001, Maedche 2003, HP Labs 2004].
- ***On-demand Services***: Services must be discovered and invoked as and when they are required; irrespective of location [WebMethods 2003].
- ***Device Independence***: Any device, irrespective of its capabilities, must be capable of joining the network, which may range from high-end personal computers to resource-limited sensors.

The following section describes how these requirements have been addressed using the DiSUS protocol.

3.3.2 DiSUS Overview

The DiSUS protocol implements three main components: the P2P interface; SISM and application specific peer services. Using these components DiSUS enables devices to publish and discover services and evolve and learn the different vocabularies used by different device and service manufacturers. Irrespective of the device's capabilities each device must implement the P2P interface, however they may chose to implement any, all or none of the

remaining components depending on its capabilities. Figure 3.2 illustrates two types of device – a Specialised Networked Appliance, defined as a device that supports high-end capabilities such as a personal computer; and a Simple Networked Appliance defined as a device with limited capabilities such as a sensor.

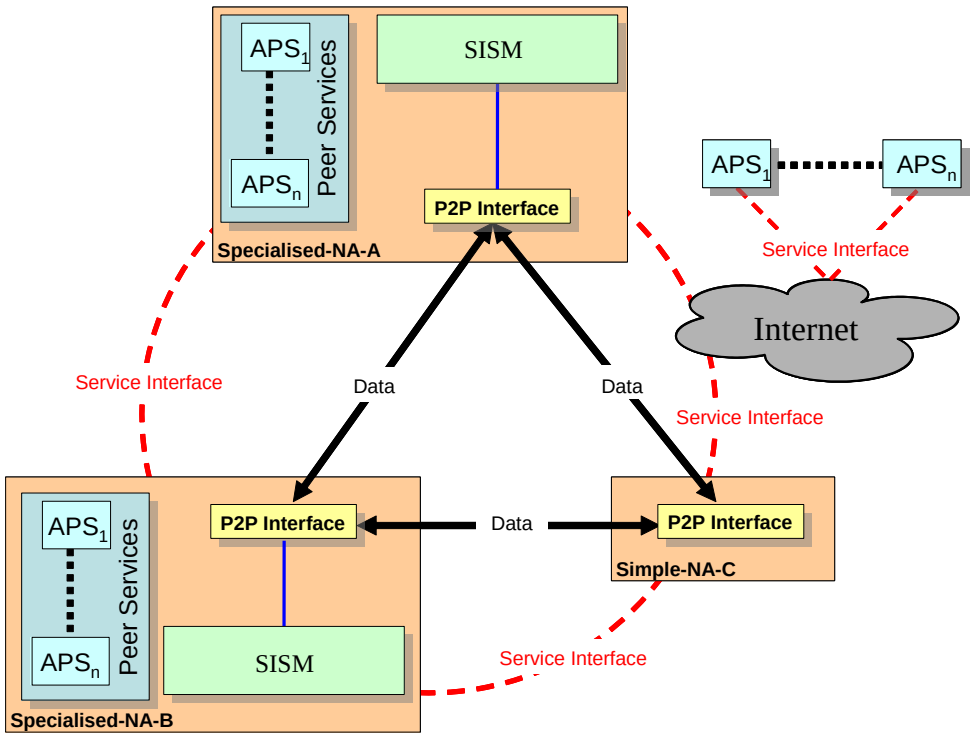

Figure 3.2 Distributed Semantic Unstructured Services

A Specialised Networked Appliance has the ability to host services, store and evolve semantic information used to describe and discover services, as well as propagate service requests within the P2P network. A Simple Networked Appliance by definition does not have these capabilities. This type of device joins the network, propagates queries and invokes discovered services without having to provide any services of its own. This is an important requirement that enables any device, irrespective of its capabilities, to effectively interact within the environment. Figure 3.2 illustrates two extremes that describe both devices that are highly capable and those that have limited capabilities, however it is envisaged that there will be a myriad of other possibilities between these extremes. In the following section, UML models are presented to illustrate how the key functions provided by the DiSUS protocol operate.

3.3.3 The DiSUS Protocol Design

The Activity Diagrams presented in this section illustrate the DiSUS protocol in NASUF. These models describe how devices execute the start-up procedure; how device capability and semantic models are created; and how peer advertisements are created, published and discovered within the network.

When a device is initially switched on it executes a start-up procedure to connect it to the network. The start-up procedure is illustrated in Figure 3.3.

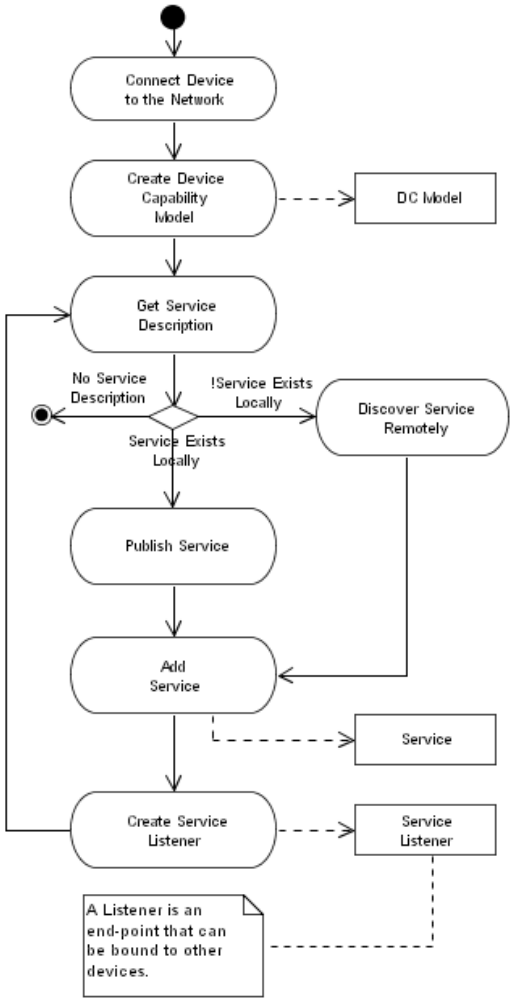

Figure 3.3 Start Device

The device is initially connected to the P2P network and a device capability model is created. The capability model captures four main capabilities used within NASUF – these are Bandwidth, Memory, CPU usage and Power. However custom capabilities may be added that are deemed important to the device manufacturer, such as screen resolution and dichotomous variables like "display in use". This model is defined using a profile, which contains several components relating to each capability. Within each component a set of attribute-value pairs are used to rank the capability defined as the Status Rating, Status Assessment, Importance Rating and Importance Ranking – a more detailed description of these are presented in Section 4.3 on page 86.

Each device publishes the capabilities it supports in order to allow devices to first determine whether it can effectively execute the services it provides. Figure 3.4 illustrates how capability models are created in NASUF.

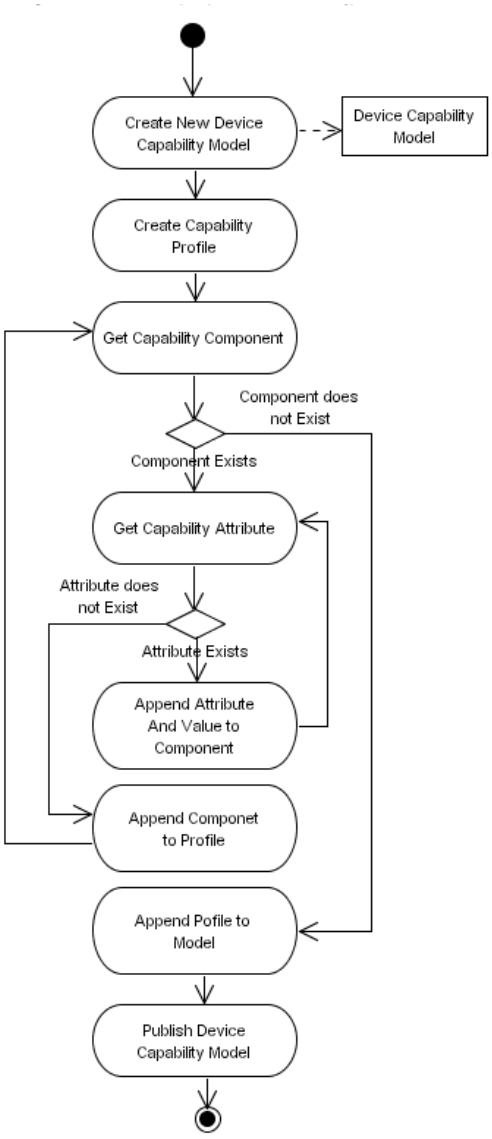

Figure 3.4 Create Device Capability Model

Once the capability model has been created each of the peer services the device uses are added to the DiSUS Manager and a listener for each service is created. If a device explicitly implements a service locally, then it is used otherwise an attempt to discover the service remotely is made. This feature is implemented on Specialised Networked Appliances only, because Simple Networked Appliances do not offer any services of their own. A device only needs to describe its capabilities to the outside world if it provides a service. Each service is created and started, before its advertisement is published – there is no point publishing the service if it cannot be started. This process is illustrated in Figure 3.5.

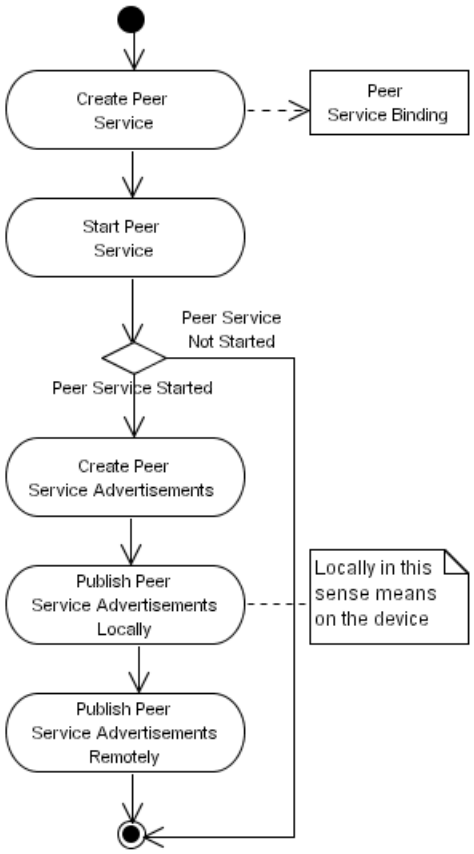

Figure 3.5 Publish Service

Peer service advertisements are created and published locally and remotely within the network. Locally, in this context, means advertisements are published on the device – this allows a generic discovery mechanism to be used that can find services either on the device itself or within the network. This enables NASUF to move away from centralised registries such as JINI [JINI Technology 2005] and UDDI [Oasis 2005] and ensures that there is no central point of failure – if a device becomes unavailable you only lose the services that device provides.

Within NASUF services are described using three advertisements – the Service Class Advertisement, the Service Specification Advertisement and the Service Implementation Advertisement. The Service Class Advertisement contains high-level information such as service provider and device information. It also contains the Service Profile, which describes the capabilities the service provides using semantic ontological structures, which are used for semantic service discovery. The Service Specification Advertisement describes the service bindings supported by the device. It also contains the Service Process Model, which groups the capability descriptions described in the Service Profile into Atomic Processes, which are used as semantic wrappers that map to signatures defined within the service interface. This is discussed in more detail in Section 4.4 on page 92. The Service Implementation Class defines

the implementation details for a particular Service Specification Advertisement. This advertisement contains the Service Grounding, which contains Atomic Processes that link the Atomic Processes in the Service Process Model with implementation specific signatures defined in the service interface. Each advertisement is linked using a unique ID. The process used to create these advertisements is illustrated in Figure 3.6.

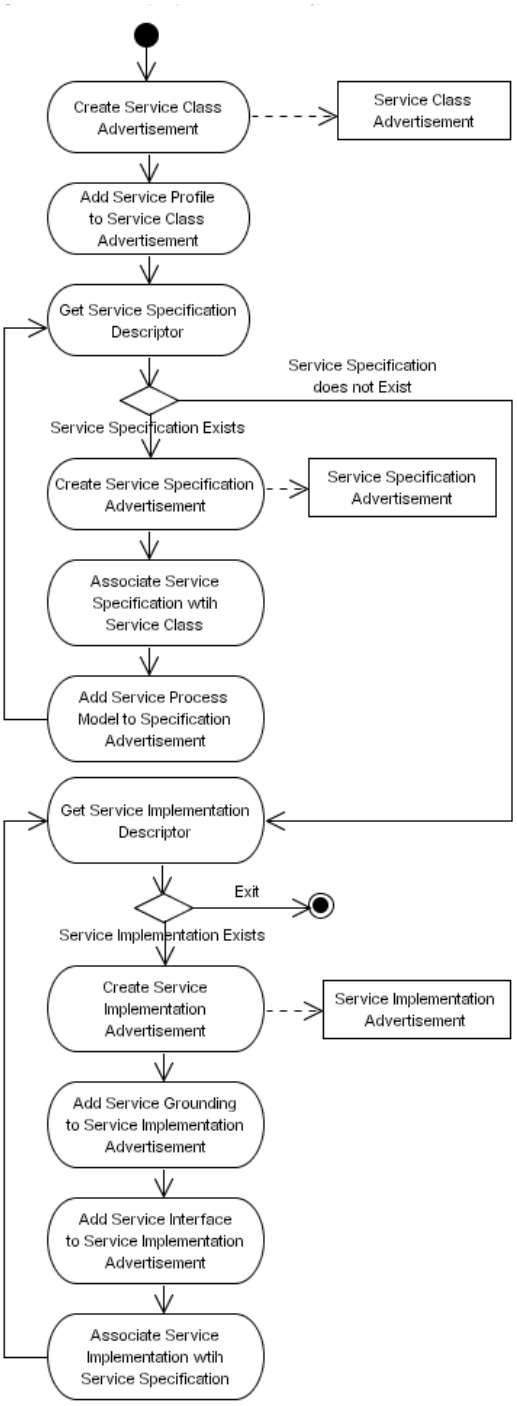

Figure 3.6 Create Peer Service Advertisements

Services are discovered and used by devices in NASUF using two methods. The first method relates to the secondary services that comprise the framework. Secondary services are pre-determined and the vocabularies used to discover these services are known by devices beforehand. The name of the service is matched against the advertisements stored within the device's advertisement cache. In this instance the name element contained within the advertisement is extracted and compared with the name defined in the service request. The process used to discover service advertisements is illustrated in Figure 3.7.

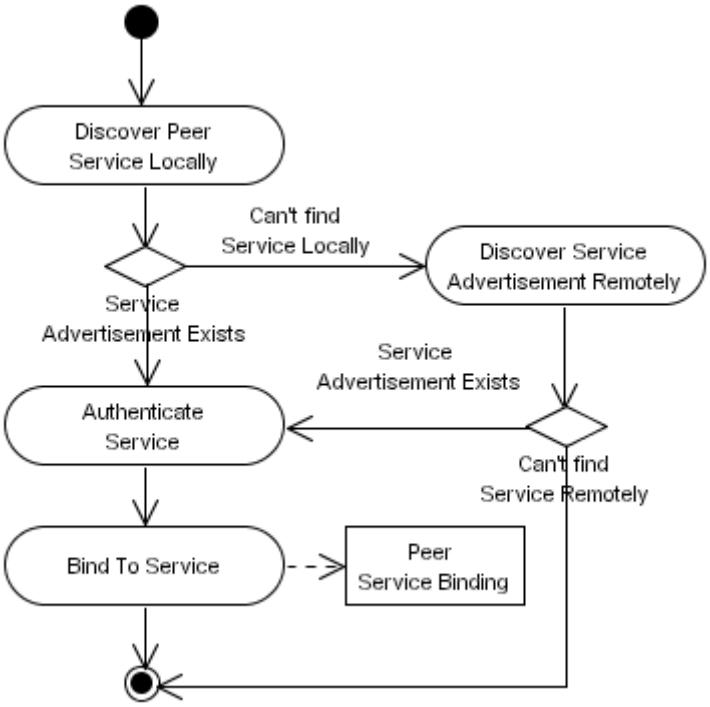

Figure 3.7 Discover Peer Service

The second method relates to application specific services. This type of service is ad hoc and the service name or its capabilities are not known beforehand. Within this project we address this problem using semantic descriptions to describe service requests and service descriptions in terms of the capabilities they support. This process is illustrated in Figure 3.8.

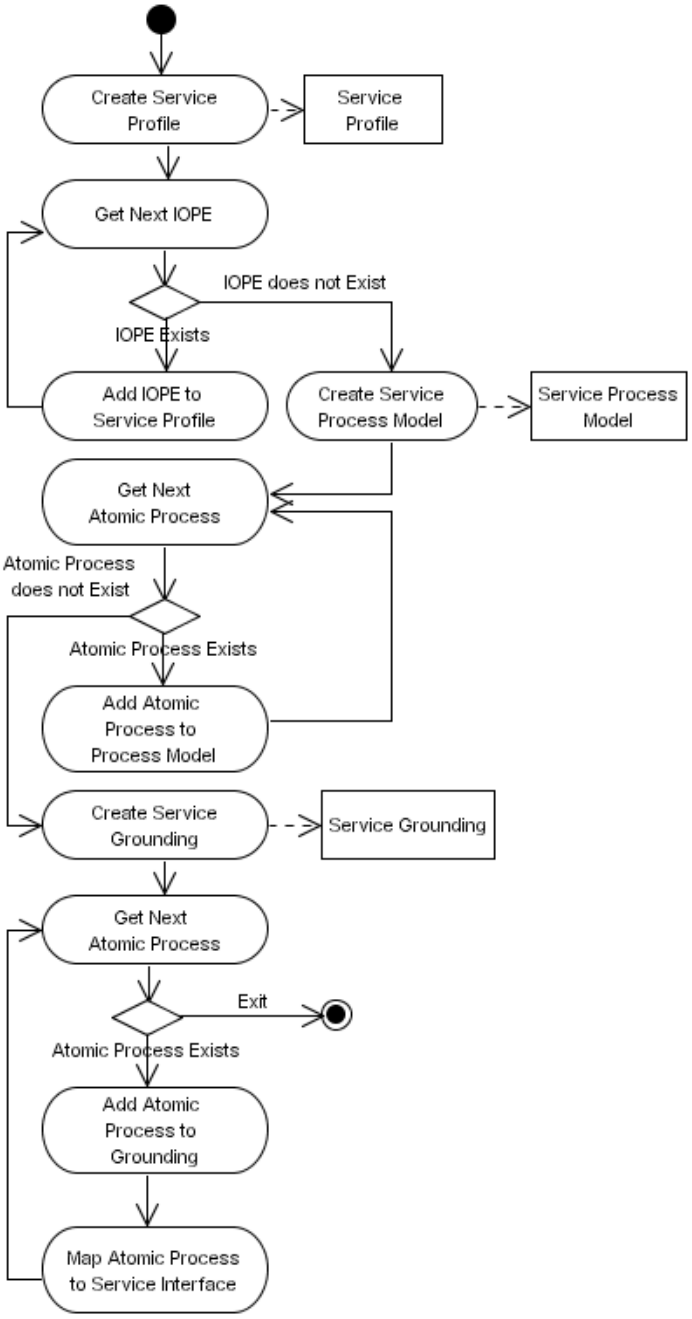

Figure 3.8 Create Semantic Models

The service request is matched against semantic descriptions contained in the service advertisements and a match is found if the capabilities described in the service request match the capabilities described in the service advertisements. For a full list of UML diagrams for the DiSUS Manager in NASUF see Appendix A, B, and C.

3.4 Summary

This chapter presented our framework. It provides an overview of all the services that comprise NASUF and describes the minimum requirements needed to allow devices of varied

capabilities to join the network and interact with other connected devices. It describes how our framework is capable of allowing devices to be dynamically distributed and discovered within a P2P network to form high-level compositions.

In the following chapter our framework is extended to include the secondary services that comprise the NASUF middleware architecture, which were presented briefly in Section 3.2. Secondary services sit on top of the DiSUS Manager. This section explains that devices do not have to explicitly implement these services, however if a device chooses not to they must be discovered remotely and bound to before the device is classed as a fully operational NASUF device. In the following chapter we describe how this is achieved and what functionality each secondary service provides.

Chapter 4

4 Framework Secondary Services

4.1 Introduction

In the previous chapter we presented our framework that each device must implement. In this chapter we describe the optional secondary services devices choose to implement. These secondary services allow application specific services, such as audio and video, to be semantically described and provide mechanisms to automatically resolve terminology differences between vocabularies used. Secondary services also provide mechanisms to enable devices to self-adapt and allow application specific services to be dynamically composed. This allows application specific services to be discovered and combined with other services based on the "what" part of the composition rather than the "how". Furthermore secondary services provide mechanisms to determine how well a device can execute a service before it commits to using it, providing a rudimentary cost metrics.

In this Chapter we present the Distributed Emergent Semantics (DistrES) service, the Device Capability (DeCap) service and the Semantic Interoperability and Signature Matching (SISM) service. These are services provided by the NASUF framework that allow device functions to be semantically described and discovered, the capabilities of devices to be assessed in terms of how effectively devices can execute services; and services that allow devices to be automatically composed, managed and self-adapted based on environmental changes.

4.2 Distributed Emergent Semantics (DistrES)

Within this thesis one of the main contributions is the use of ontologies for the purpose of service descriptions and dynamic service composition. This approach brings with it additional challenges because it is difficult to constrain how different device manufacturers develop and use ontological structures. Through peer collaborations devices need to understand the different terminology used by different devices and dynamically evolve localised knowledge structures to extend or reify concepts they already have [Fergus 2003b]. This being the case mechanisms need to be developed that can evolve such structures and bridge the gap between concepts that are semantically equivalent but syntactically distinct. Such mechanisms enable semantic interoperability between different concepts and provide a high-level of flexibility that does not constrain how services are described [Fergus 2003b].

The DistrES algorithm is designed to discover semantic information provided by devices connected to the network and merge the results with existing knowledge structures. Devices initially have knowledge that support the vocabularies used to describe their own services, however knowledge structures are extended over time to include the vocabularies used by other devices to describe similar concepts. A simple scenario is illustrated in Figure 4.1.

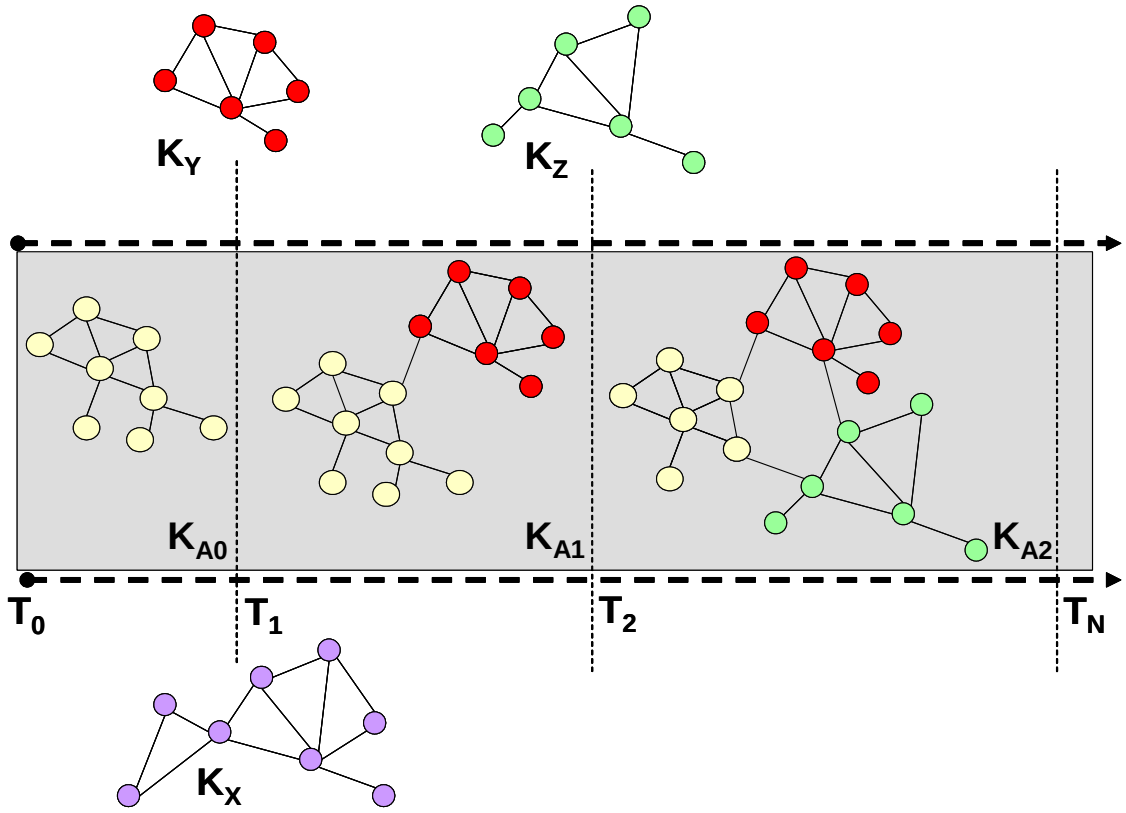

Figure 4.1 Evolving Knowledge Structures over Time

The basic assumption is that a device will have a limited amount of information and will evolve its internal knowledge structures over time by interacting with devices in the network – K_{A0}, represents a device with limited knowledge. At T_1 two information structures are presented to the device, labelled K_X, and K_Y. The device determines that K_Y is a knowledge structure that matches a query it has propagated within the network. The K_Y structure is identified as the most successful structure based on several responses received from the network. The success of this structure is determined by statistically evaluating all response knowledge structures received after T_0 and extracting the common patterns found within those responses to produce the K_Y structure. This new structure is merged with K_{A0} to become K_{A1}.

At T_2, the device propagates a query to the network. During this cycle K_Z represents a structure that matches the query. In this case, the structure K_Z is identified as the best information structure based on the number of common patterns found in all the responses received after t = 1. This new structure is merged with K_{A1} and becomes K_{A2}.

It is possible that this process leads to isolated information structures within the device's knowledge base, which are detached from the root node. However over time these structures will form connections to other knowledge structures as the device's information evolves. This is illustrated at t = 2 in Figure 4.1. When K_Z is merged with the current information structure a relationship is found between the information structure at t = 0 and the information structure merged at t = 1. As a result this technique is able to determine relationships between fragmented information structures and perform appropriate merges to connect them to the main structure. This is possible because of the classification mechanisms used to construct ontologies where classes may have many relationships with other classes – explicitly placing a relationship between two concepts automatically links them together. How information is structured will be the deciding factor as to whether concepts are linked with main structures or remain isolated. This will be dependent on the general consensus, *i.e.* if the majority believe that a concept *dog* is a subclass of another concept *reptile* then these will be explicitly linked, however if the majority correctly believe that the concept *dog* is not a subclass of the concept *reptile* then *dog* may remain isolated from the concept *reptile*. It depends on the scope of the domain being modelled and how concepts are generally constructed.

This mechanism is designed to enable a device's ontology to be evolved as it moves through time and interacts with other devices within its environment. The following section describes the requirements needed to implement the DistrES algorithm and explains in detail the sub processes it uses.

4.2.1 The DistrES Algorithm Requirements

In this thesis conceptually merging information structures is based on general consensus. For example if nine out of ten people state that a concept *Alsatian* is a subclass of another concept *Dog* then these terms including their associated relationships will be described in the optimal structure because there is a general consensus agreement. The success of concept proliferation is dependent on the consensus percentage. For example if 51% of devices believe that *Alsatian* is a subclass of *Dog* then these concepts will form part of future structures because the majority believe that this ontological structure is true. The converse of this is that if for example only 30% of devices believe this to be true then the chance of this structure appearing in future structures is decreased and the structure will eventually vanish. Quantifying this is difficult because how successful an ontology structure is will be dependent on the number of concepts that exist; the number of devices there are within the network; and the global consensus on how structures are created. The DistrES algorithm is a mechanism that embraces this uncertainty and enables ontological structures to be evolved based on the environmental conditions at any given time. It can adapt to ontological and general consensus

changes. In order to achieve this, the DistrES algorithm is required to create, extract and merge information within an ad hoc network environment. Consequently the algorithm must address the following requirements:

- ***Knowledge Structure***: Knowledge structures must be nodes sub-classed taxonomically from some root node. However, fragmented structures may exist but must be merged into existing structures as the device's ontology evolves over time. The structure of information must be represented in an open standard format (electronically readable) and must be searchable (in knowledge base) and be fully editable.
- ***Targeted Knowledge Discovery***: Devices must have the ability to evolve existing information structures by propagating queries within the network about subsections of their ontology they wish to extend, *e.g.* "Movie". It is the DistrES algorithm that determines when and what structures to evolve based on any ambiguities that may be encountered.
- ***Extraction Engine***: When a device processes a query and determines that it has relevant information structures, it has to extract this information from its ontology and return it back to the querying device. Although this is the function of the knowledge base, DistrES must define what subsection of the concept needs to be extracted. This is an application specific function, which will be dependent on the device and how rich the ontological structure should be. For example a mobile phone may only require a concept that has a depth of three or less (subclasses), whilst a PC may require a richer representation that has a depth of ten. This is important because the depth of the concept will determine its size – the bigger the structure the more memory and processing is required. Consequently the Extraction Engine must have the ability to control this process.
- ***Statistical Pattern Extraction***: Within the network a querying device may receive several responses and the structure of the information within these responses will differ. There is no centralised control and no assumptions can be made about the level of expertise creators of knowledge will have. As such information structures need to be evolved based on general consensus, which must be determined by evaluating ontological structures in all responses received. This is achieved using statistical analysis [Rumsey 2003], which extracts patterns from ontological structures being processed until an optimal solution is created and merged within the device's local ontology.

- ***Merge Engine***: When a device receives an optimised response from the Statistical Pattern Extraction Engine, this information needs to be merged with the device's existing ontology.

4.2.2 The DistrES Algorithm Overview

The DistrES algorithm extracts and merges information from ontologies and evolves information structures to produce best solutions based on a general consensus. This is achieved using the Extraction Engine (EE), the Statistical Pattern Extraction Engine (SPEE) and the Merge Engine (ME).

Extraction Engine: Devices process queries propagated within the network and extract the name of the concept. The concept name is used to query the device's ontology to see if the concept exists. If it does the process begins by extracting all the dependents and for each dependent found, the Extraction Engine retrieves all the relationships that exist between the concept and all its dependents.

Statistical Pattern Extraction Engine: Devices propagate queries containing concepts they wish to evolve, to other devices within the network. This may result in the device receiving several responses which contain ontological structures that are subjective based on the creator's own point of view. This leads to structural and possibly lexical variation between all responses received. This research aims to address this problem using the Statistical Pattern Extraction Engine (SPEE). The SPEE extracts structural patterns based on commonalities found within all responses and produces an optimal ontological structure that is said to capture the general consensus.

Figure 4.2 illustrates a subset of a device's ontology (C1) and three responses (R1 – R3) representing the results the device has received from the network based on a query it submitted.

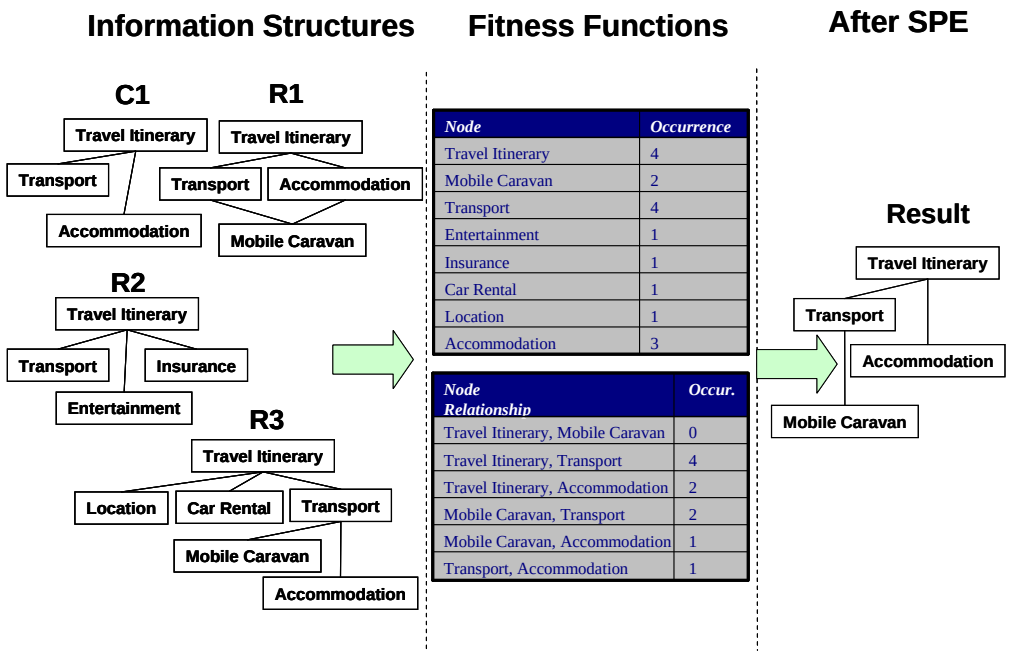

Figure 4.2 Statistical Pattern Extraction Engine

It is clear that although structurally R1 – R3 are different, there are commonalities within the structures that are apparent in them all. For example, the nodes "Travel Itinerary" and "Transport" have a direct relationship between each other in all the structures. The SPEE determines that this is a common pattern by calculating the number of times this relationship occurs in all structures being processed – if there are four structures and four occurrences of the relationship then it is said to be common to all structures, *i.e.* 100% consensus, and based on the general consensus it should appear in the optimal structure.

In contrast Figure 4.2 also illustrates that the nodes "Entertainment", "Insurance", "Car Rental" and "Location" are low scoring nodes because each node appears in one structure only. The SPEE classes these low-scoring nodes as uncommon, *i.e.* 25% occurrence, and the probability of these nodes appearing within the optimal structure is greatly reduced.

Initially all the unique terms, including the relationships that exist between terms are derived from the device's extracted concept (C1) and all the responses (R1 – R3). Using two fitness functions, the SPEE decides which terms and which relationships will appear in the optimal structure – terms and relationships with the highest fitness values are extracted and used to rebuild the optimal structure. Fitness functions are configurable. The higher the fitness function the more specific the extraction process is. The lower the fitness function the more general the optimal structure will be. In this sense low fitness functions will enable optimal structures to be created that have low-scoring nodes, whilst high fitness functions provide a mechanism that filters concepts and relationships that are not generally used. Setting fitness functions may be dependent on a number of factors such as accuracy, concept size and

processing time. For example a mobile phone, which is process and memory restrictive, may require specific information to ensure that less common information is not included. This will enable the device to minimise the size of the ontological structures that need to be stored. Furthermore it will also decrease the amount of processing required.

The first fitness function places all the unique nodes, found within all structures, into a collection, *e.g.*

[Travel Itinerary,

Mobile Caravan,

Transport,

Entertainment,

Insurance,

Car Rental,

Location,

Accommodation]

These nodes are given a fitness value based on the number of times a node appears within all structures, which we call term frequency. For example, the node "Travel Itinerary" is given a fitness value of four because it appears once in all structures. The second fitness function places all the possible relationships into a collection that may exist between any two nodes, *e.g.*

[Travel Itinerary|Mobile Caravan,

Travel Itinerary|Transport,

Travel Itinerary|Accommodation,

Mobile Caravan|Transport,

Mobile Caravan|Accommodation,

Transport|Accommodation]

The fitness value of each relationship is calculated based on the number of times a relationship appears within each structure, which we call relationship frequency. For example the relationship between "Transport" and "Accommodation" is given a fitness value of one because the relationship appears in only one of the four structures.

Once the SPEE has a list of ranked nodes and relationships, it extracts the top scoring nodes and the top scoring relationships and uses them to rebuild the optimal structure. For

illustration purposes the 'Result' structure in Figure 4.2 is generated by using the top four nodes and using the top three relationships. These are arbitrary values, which in a real-world situation, will be application specific. As described above, several factors decide what these values should be, *i.e*. accuracy, concept size and processing time. This means that the most optimal structure would be represented by the following nodes:

[Travel Itinerary,

Mobile Caravan,

Transport,

Accommodation]

And the following relationship collection:

[Travel Itinerary|Transport,

Travel Itinerary|Accommodation,

Mobile Caravan|Transport]

These are used by the SPEE to construct a new ontology structure, which is then merged with the devices existing ontological structures using the Merge Engine.

Merge Engine: The Merge Engine iterates through the ontological structure produced by the SPEE and attempts to merge the nodes and relationships with existing knowledge structures. This process begins by iterating through all the nodes found within the structure and determining whether the node already exists in the device's knowledge base - if the node does not exist, a new node is created and inserted into the knowledge base. Once this is complete the process is repeated for all the relationships that exist between the nodes in the structure. This is explained in more detail below.

4.2.3 The DistrES Algorithm Design

This section presents several UML Activity Diagrams that illustrate how the DistrES algorithm has been designed. These models describe how concepts are searched for, extracted, evolved and merged with existing knowledge structures.

When a device is trying to determine if a relationship exists between two terms the DistrES algorithm begins by trying to find a semantic relationship. For example an *Alsatian* is a *subclassOf Dog*, consequently these terms could be used interchangeably and we could infer that they, to a certain extent, mean the same thing. In this instance the two terms are *Alsatian* and *Dog* and the relationship is *subclassOf*. If a *subclassOf*, relationship can be found between the two terms then the concept surrounding the terms and the relationships that link

them are extracted and returned to the service requester. Any ambiguities between terms, triggers the evolutionary process and in this instance the ambiguity could be because the device does not have enough knowledge to relate the concepts. Thus the device tries to evolve its existing knowledge structures in an attempt to determine whether a relationship exists in other knowledge structures provided by devices in the network. This process is illustrated in Figure 4.3.

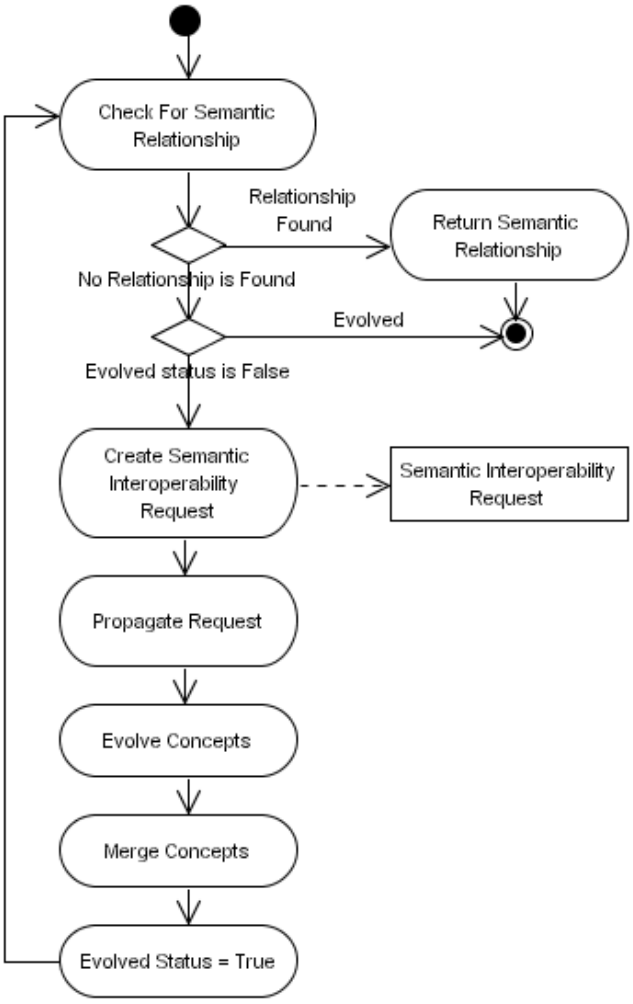

Figure 4.3 Semantic Interoperability

If a relationship cannot be found and the two terms have not been evolved then DistrES creates a semantic interoperability request that contains the two terms and propagates it within the network. This request is processed by other DistrES services within the network and used to determine if the device's ontology has a relationship that links the two terms together. This process is illustrated in Figure 4.4.

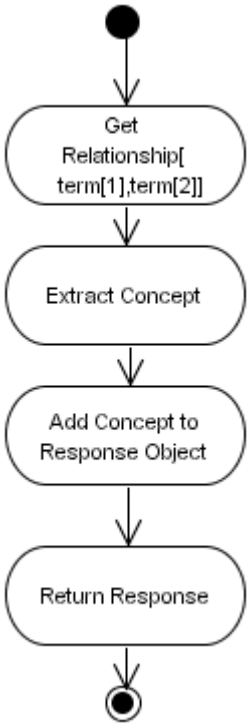

Figure 4.4 Extracting Ontological Structures

If at least one relationship is found the concept surrounding the terms is extracted and added to a response object, which is then returned to the querying device. Any responses returned to the querying device are evolved using the SPEE to produce an optimal structure. This process is illustrated in Figure 4.5.

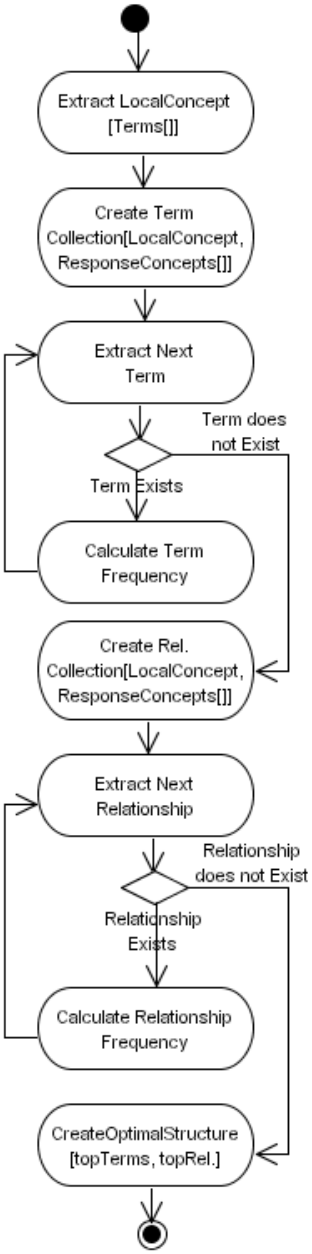

Figure 4.5 Evolving Ontological Structures

All the unique terms in the concept, including all the responses received from other devices within the network, are extracted and placed into a collection. This process loops through the collection and for each term it checks how many times it appears within all the structures being processed. This results in a term frequency value. Once all the terms have been assigned a corresponding fitness value the same process is performed for all the unique relationships that exist between the nodes. Again the relationship frequency is calculated resulting in a relationship frequency value. Once this process is complete, the top *n* fittest nodes and the top *n* fittest relationships are used to create an optimal structure. In this instance *n* is an application specific value defined by the device depending on its capabilities. This

value is used to constrain the size of the optimal structure created, which is merged with the devices existing knowledge structure. This process is illustrated in Figure 4.6.

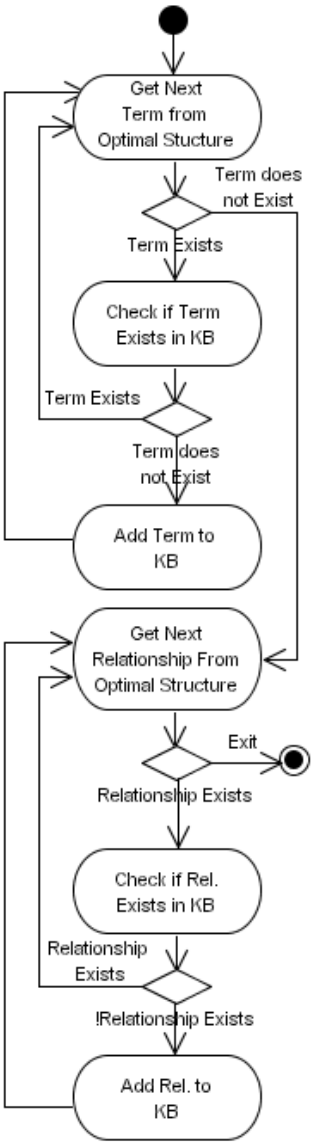

Figure 4.6 Merging Ontological Structures

Each of the terms that exist in the optimal ontology structure is checked to see if it already exists in the device's local ontology. If the term does not exist it is added to the device's local ontology. This iteration process continues until all of the terms have been processed. When this occurs the process is repeated for each of the relationships that exist within the optimal structure. If a relationship does not exist in the device's local ontology a new relationship is created. Again this process continues until all the relationships in the optimal structure have been processed.

Once the structure has been merged the status of the evolution process is set to true – this stops the algorithm from continually looping as we only want to try and evolve a set of terms once at any give time.

4.3 The Device Capability (DeCap) Service

When services are discovered and matched this may result in several candidate services that provide the same functionality. Services provided by devices that best match the device capability requirements, as described in the service request, must be selected. For example a typical home environment may provide several 'Visual' display services capable of processing streamed data – typically devices that provide the best quality of service will be selected, *i.e.* a 'Visual' service provided by a TV may be selected instead of a 'Visual' service provided by a mobile phone to view a DVD Movie. However, if the mobile phone is the only device available, then an intermediary service may be discovered to transcode the DVD movie into a format that can be readily processed by the mobile phone.

Consequently each device that joins the network within NASUF must describe the hardware, software and networking capabilities it supports, including any other capabilities deemed important to the device manufacturer. Figure 4.7 illustrates the process used that matches the Device Capability Profile (DCP) described in the service request, with the Device Capability Model (DCM) used to describe a particular device's capabilities [Mingkhwan 2005].

In this example a multimedia player begins by creating a DCP and adding it to a service request before propagating it within the network. An audio device receives the message and begins by checking to see if it provides a service matching the requirements defined in the service request. If a service is found, the device determines if its DCM equals or surpasses the DCP.

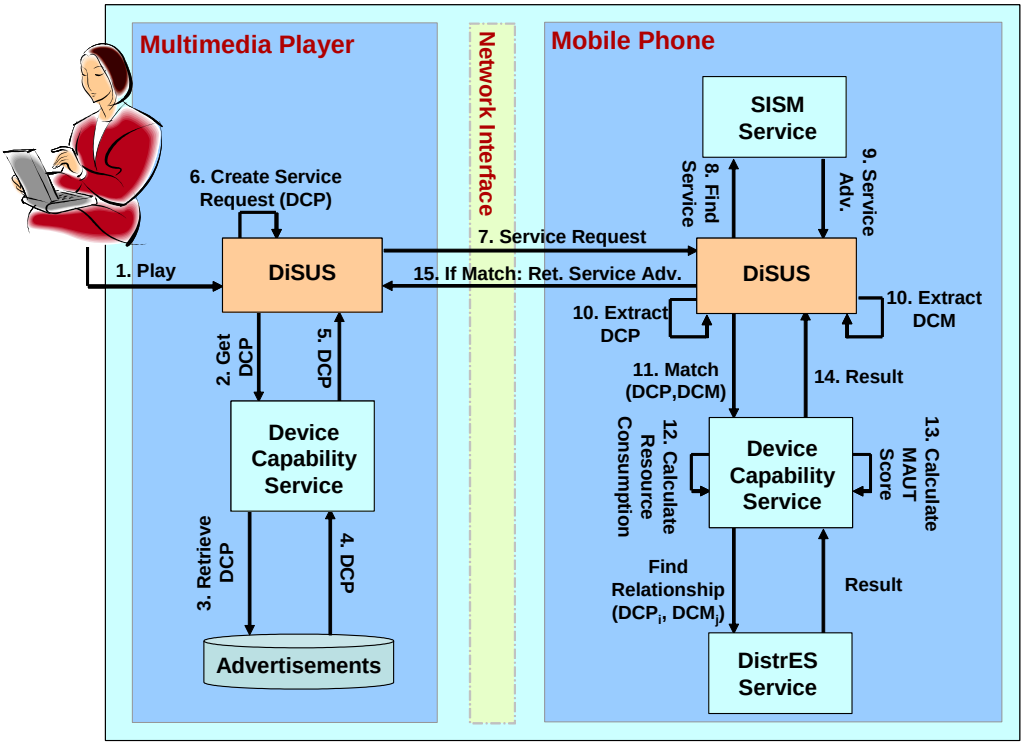

Figure 4.7 Device Capability Matching Service

This is achieved by extracting the DCP from the service request and the DCM from the devices persistent storage. Using the Device Capability (DeCap) service, the two models are passed to a matching algorithm that calculates the overall capability of the device – if the result of the DCM is equal to or greater than the result calculated for the DCP then the device is said to satisfy the capability requirements defined by the service requester. In this instance the device returns the service advertisement to the multimedia player. Again like service requests and service descriptions, NASUF does not place any constraints on the vocabularies used to describe quality of service parameters. If the terms found in the DCM and the DCP differ the matching algorithm uses the DistrES service to determine if a semantic relationship exists that links the terms together.

4.3.1 The DeCap Service Requirements

Device capabilities are determined by calculating the sum of all quality of service parameters used that capture the software, hardware and networking properties supported by the device, including custom defined parameters. In order to achieve this, an algorithm is required to calculate the quality of service capabilities described in service requests and device capability models that address the following requirements:

- Process QoS Parameters: Quality of service parameters must be used to capture the software, hardware, networking and custom capabilities a device supports. These parameters must be defined and inserted into the service request before it is

propagated within the network. These parameters must also be used to describe the capabilities the device actually supports and inserted into the DCM.

- Assign Parameter Rating and Status: For each parameter defined in the DCP, an importance rating must be assigned indicating how important it is in relation to all the parameters being used. Furthermore a status rating to indicate how well the device conforms to that parameter, such as 100 for excellent, 50 for average and 0 for poor. The importance rating and the status rating must be multiplied to give a weighting value for a particular parameter being processed. The weighting value is created and used by the DeCap service and indicates how well a device supports a parameter. This value is added to the overall capability score, which is used to determine whether the score produced for the DCM is equal to or greater than the capability score calculated for the DCP.
- Calculate Overall Quality Rating: Each of the parameter weightings must be added together to give an overall weighting for the capabilities the device supports.

These requirements enable devices to determine whether a device that provides a particular service is capable of executing the service in conformance with the requirements defined in the DCP.

4.3.2 The DeCap Design

The matching algorithm used within the DeCap service uses two calculations to calculate the current resource load and the load required to execute the service. DeCap begins by calculating the resource expense incurred when the service is executed by adapting the formulas defined in [Kumar 2003, Liu 2004]. The formula defined in (1) calculates the percentage of a resource required, where a resource *r* offers a service *s* that requires $ac_{s,r}$ units of some total resource value tr_r.

$$resc_{s,r} = \frac{ac_{s,r}}{tr_r} \quad (1)$$

This formula allows the DeCap service to determine what percentage of some resource will be used given the total value of the resource available. For example, if a service requires 1 megabyte of RAM and the device provides a total of 32 megabytes, then the service is said to require 3.1% of that total.

However it is not enough to only calculate the value for the resources needed to execute a service. The DeCap service needs to determine if the device is overloaded by calculating how much of the available resources on average are used by the device. For example, if the device on average has 75% CPU utilisation, we can infer that the device may struggle to take on the

increased computational overhead if our service is to be executed. The cut-off value used to determine when a capability is no good is application specific and dependent on the task in hand. For example if the service is a transcoding service then the application may state that CPU utilisation should be no more than 10% because the CPU required to perform the transcoding will be approximately 90%. Conversely a service that processes simple commands such as "Light switch on" may require minimal computation thus a device that has 75% CPU utilisation will be capable of incurring the additional computational overhead without causing adverse effects.

Furthermore it is possible that the quality of service will be affected because the computation may be shared across a large number of processes. When this is the case, DeCap calculates the overhead for each resource the service requester deems important and compares it to the desired capability defined in the service request. The DCS achieves this using a technique called the Multi-Attribute Utility Theory (MAUT) [Kumar 2003, Liu 2004]. This algorithm is implemented in DeCap and is used to produce an overall capability score for some device *D* given the attributes defined in the device's DCM. This formula is defined as,

$$DCScore(D, DCM) = \sum_{i=1}^{d} cw_i(DCM) \cdot D(v_i) \qquad (2)$$

where *DCScore* is the overall capability score for device *D* according to the device capability model *DCM*, *d* is the number of capabilities for the type of device, $cw_i(DCM)$ is the importance rating of attribute *i* according to devices *DCM*, and $D(v_i)$ is the status rating for attribute *i*. The importance rating describes how important a given attribute is in relation to all the attributes used, *e.g.* the CPU attribute may be the second most important attribute with an importance rating of 30, which means that the CPU is considered three times more important than an attribute with an importance rating of 10. The status rating describes how well the device supports a particular attribute, *e.g.* a device may have "Excellent" for its CPU attribute, which may equate to a value of 75 – therefore calculating a capability score for CPU, could be achieved by multiplying 30 * 75 which is equal to 2250.

Given the two formulas, the device calculates the service ratings programmatically by estimating the average attribute values from the operating system itself and assigning the appropriate status rating. For example, if the device uses on average 25% of its CPU when the required service is executed we may assign the CPU_Load a status assessment of "Excellent" with a status rating of 75.

The equation defined in (3) amends the MAUT formula to take into account the current resource load and the load required to execute a service. In this instance the *DCScore* and the

$resc_{s,r}$ are added to give a combined resource load value, indicating whether the device can effectively execute a service it provides.

$$DCScore(D, DCM) = \sum_{i=1}^{d} cw_i(DCM) \cdot D(v_i) \cdot (1 - resc_{s,r}) \quad (3)$$

When the terms in the DCP and the DCM are processed any ambiguities that are encountered are resolved using the DistrES algorithm. When the formula in (3) is used to calculate the score for the DCM, it is compared with the score generated for the DCP. If the DCM score is equal to or greater than the score for the DCP then the device is said to be capable of effectively executing the service, whilst ensuring the quality of service is maintained. In this instance the service details are returned to the service requester.

The following models describe how the DCP is matched with the DCM. This process is illustrated in Figure 4.8.

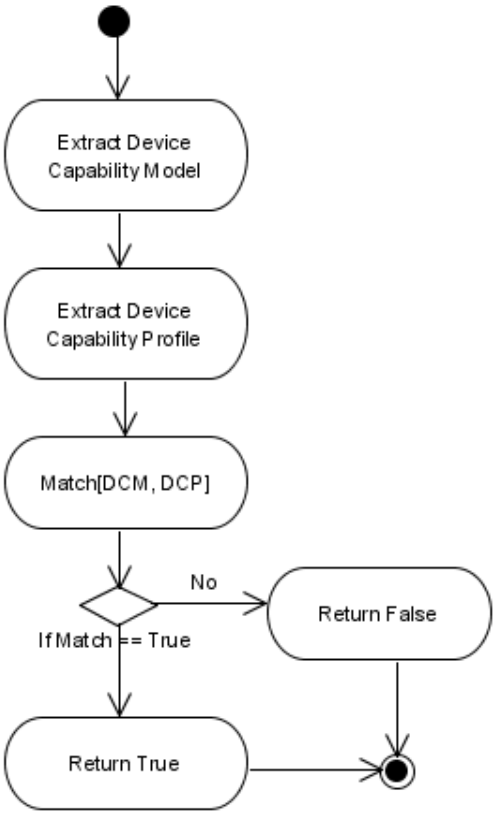

Figure 4.8 Device Capability Matching

This process begins by extracting the DCM from the device currently processing a received service request. DCMs and DCPs are Device Capability Advertisements that capture all the key hardware, software and network capabilities. The Device Capability Advertisement Object contains a device capability profile, which in turn contains a collection of component objects. Each component object contains a collection of attribute objects that describe a

capability including its associated value. The class diagram for Device Capability Advertisements is illustrated in Figure 4.9.

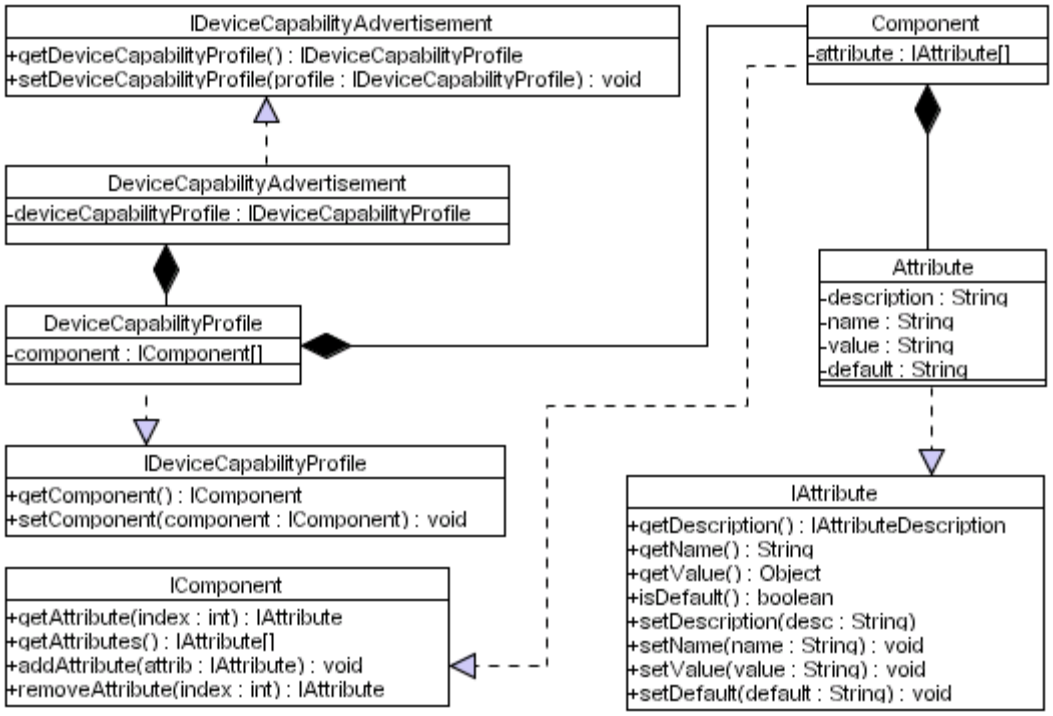

Figure 4.9 Device Capability Advertisement

Once the DCMs have been loaded, the DCP is extracted from the service request and loaded. They are passed to the DeCap Service, which returns a value of true or false indicating whether the device has the required capabilities to execute the service it provides, based on the requirements defined in the DCP.

The matching process as illustrated in Figure 4.10 loops through all the quality of service parameters used and for each parameter extracted this process retrieves the *importance rating* from the DCP. If the rating does not exist then the next parameter is extracted and the process is repeated. If it does exist the *status rating* is retrieved from the DCM. Again if the rating does not exist the next parameter is retrieved and processed. The *importance rating* and the *status rating* are multiplied together and added to the overall quality of service result. Once all the parameters have been processed the result is returned. If the returned value fro the DCM score is equal to or greater than the score for the DCP then the device is said to be capable of executing the service is provides.

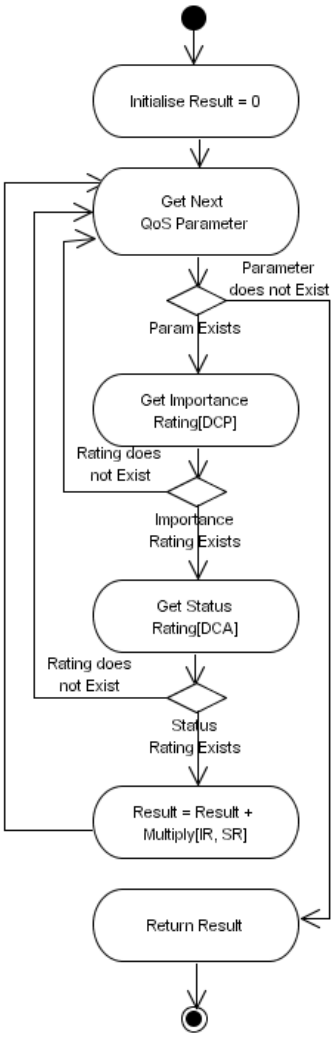

Figure 4.10 Device Capability Matching Algorithm

4.4 Semantic Interoperability and Signature Matching (SISM) Service

Within this thesis one of the key requirements is to enable devices to self-adapt and form compositions with ad hoc services. Current home networking platforms perform interoperability between heterogeneous devices by standardising the interfaces devices implement imposing pre-determined vocabularies. This technique is restrictive and is difficult to implement within uncontrolled environments. We address this limitation using a service we have developed called the Semantic Interoperability and Signature Matching (SISM) service [Fergus 2005a, Mingkhwan 2005].

SISM works by processing metadata used to describe the service and the service request, including the signatures described in the service interface. Service descriptions and service requests are described at an abstract level in terms of the Inputs, Outputs, Preconditions and Effects (IOPE) they describe, which are more commonly referred to as IOPEs [DAML

2003c]. SISM allows highly independent services offered by appliances to be dynamically composed without any human intervention. A high-level description of the possible compositions performed using SISM, is illustrated in Figure 4.11.

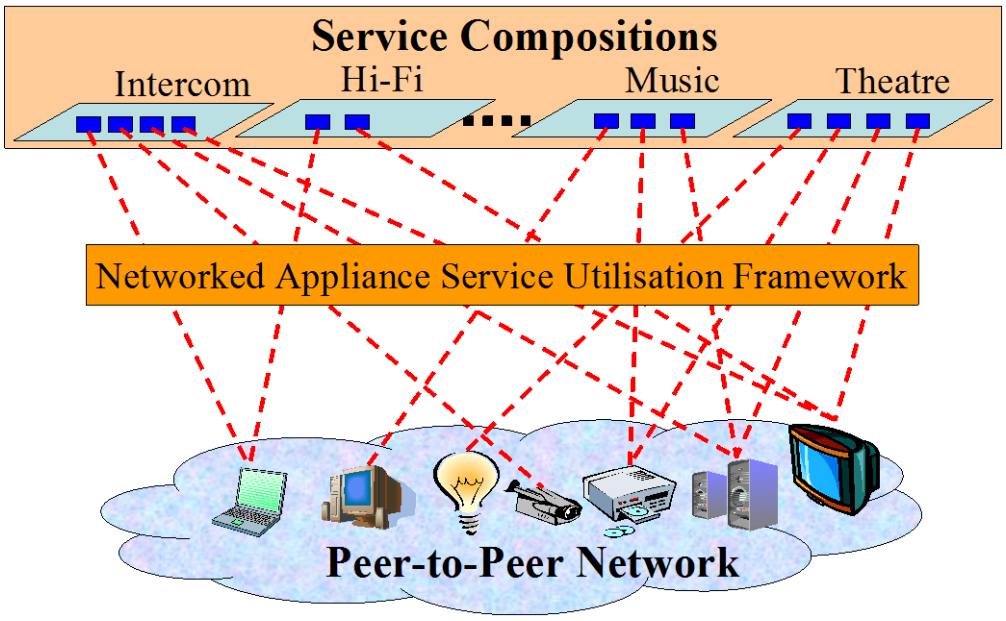

Figure 4.11 Dynamic Service Compositions between Devices

This diagram illustrates that by using NASUF the individual functions provided by networked enabled devices can be selected and composed to create high-level functions. For example by discovering all the audio services in the network, and using a microphone provided by either the laptop or camcorder, a composition of services can be combined to create an intercom system. Processing the high-level semantic descriptions of services forms the basis for this approach and is described in more detail in the following sub sections.

4.4.1 The SISM Service Requirements

Composing services poses a difficult challenge. This section describes the key requirements to be addressed in order to enable SISM to automatically achieve this without any human intervention. An algorithm is required to automatically compose services using semantic descriptions thus:

- Mechanisms need to be developed that allow the services offered by devices to be automatically discovered and dynamically composed.
- Services need to be described semantically in order to expose the capabilities they support.
- Devices must be selected to ensure a high quality of service is maintained.

The remainder of this section describes how these challenges are addressed using the SISM Service developed within this thesis.

4.4.2 The SISM Service Overview

The SISM algorithm matches IOPEs and signatures described in the service interface, which supports direct matches, indirect matches and conflict resolution. Using this algorithm, services can be matched and the service interface can be dynamically extended beyond what it was initially designed to do.

4.4.2.1 The IOPE Matching Process

The SISM Service can determine if any two terms match using a number of techniques. One possible match occurs when any two terms are equivalent, which is illustrated in Figure 4.12.

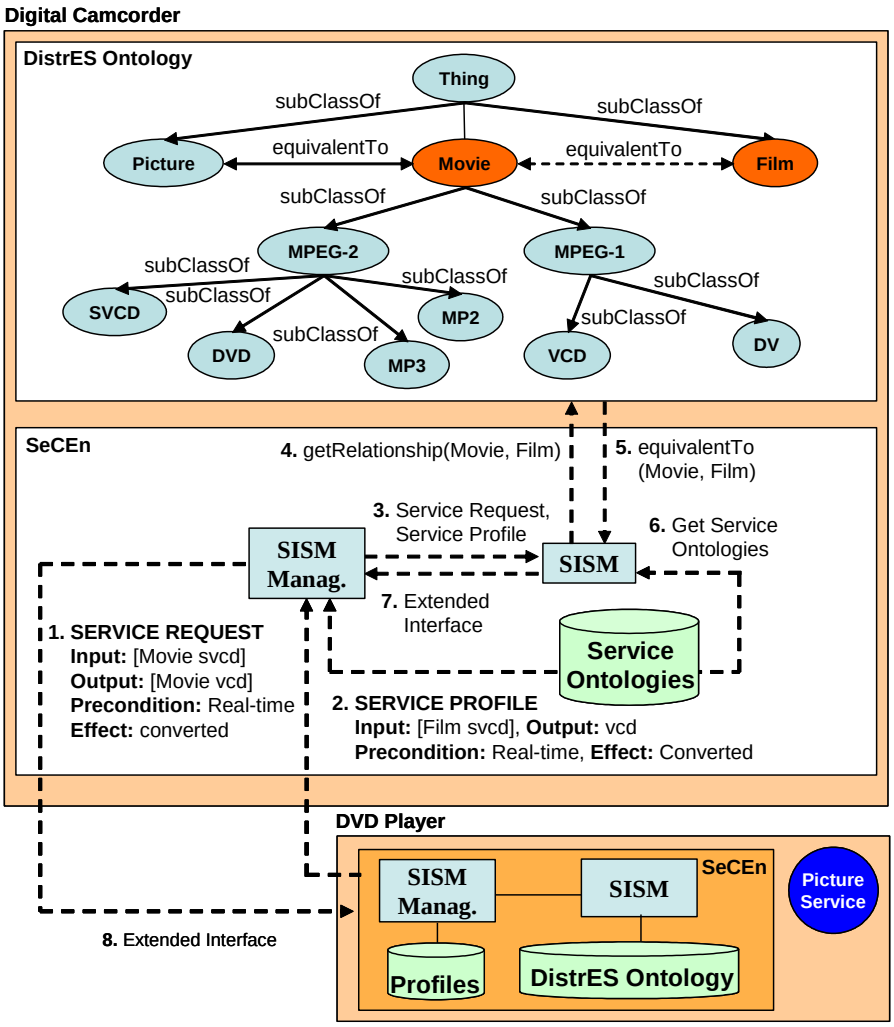

Figure 4.12 IOPE Matching performed by SISM

In this example the precondition 'Real-time' in the service request is equal to the precondition 'Real-time' in the service description. Another possible match can be achieved via subsumption. For example an input in the service request may be called "Movie" and an input in the service description may be called "Film" – if "Movie" is either a 'subclass', 'superclass' or 'equivalent' to "Film" then a conceptual relationship has been found that links

the two terms together. However this example is problematic because the term “Film” could mean “Movie” or “Slideshow”. In this instance the name of the inputs and the outputs are used to help determine the context in which the term is being used. This matching process is described below:

- If the IOPE in the service request is the same as the IOPE in the service description then this constitutes an exact match.
- If the IOPE in the service request has an ‘equivalentTo’ relationship with the IOPE in the service description then this constitutes an exact match.
- If the IOPE in the service request is a subclass of the IOPE in the service description then this constitutes an exact match.
- If the IOPE in the service description subsumes the IOPE in the service request then this constitutes a plug-in match, *i.e.* if concept *A* is a sub-concept of a concept *B* this is called a plug-in match. This is a useful matching process that loosely relates concepts that exist in the same hierarchy path. However, the distance between the two concepts need to be determined in order to establish how closely related they are. If concepts are closely related then it may be possible to interchange these concepts without altering the meaning. In this type of match concepts in the service request are typically more general than concepts in the service description. This may result in service descriptions being too specific for the service request [Paolucci 2002a].
- If the IOPE in the service request subsumes the IOPE in the service description then this constitutes a subsumption relationship. For example if concept *A* is a super-concept of concept *B* then concept *B* is subsumed by *A.* This type of match is weaker than a plug-in match in that concepts in the service request are more specific than the concepts in the service description. Although matches may occur, again it comes down to the distance between concepts – in some cases the match may be too general [Paolucci 2002a].
- Anything else fails.

If a relationship cannot be found, the unknown term is passed to the DistrES Service [Fergus 2003b] and propagated within the network. This results in zero or more semantic structures being returned that describe how the term is defined. Using statistical programming techniques, such as term and relationship frequency analysis, the structures are evolved until an optimal solution has been produced and merged into the DistrES ontology [Fergus 2003b]. Once the structure has been merged the above matching process is repeated. This process continues until all the IOPEs in the service request have been processed – if all the IOPEs are matched this constitutes an abstract match. This could potentially be time consuming. Depending on the number of responses and the size of individual concepts devices may

choose to perform this as a backend process. Consequently devices may choose to evolve concepts offline. This may ensure that the next time a device processes a service request it can better match the IOPEs in the service request and the service description.

When abstract matches are found, SISM retrieves the service ontologies [DAML 2003c], along with the service interface object and creates a table containing the matching IOPEs from the service request. A sample table may look like the one illustrated in Table 4.1.

Service Request Term	*Device DCM Term*
Movie	Film
TV	Television
Speaker	AudioDevice
MPEG-2	DVD

Table 4.1 Semantic Interoperability Table

The matched IOPEs act as keys in the table and have corresponding values, which represent the names of the IOPEs used in the service ontologies. SISM uses the DistrES ontology to resolve terminology differences, therefore the service request may refer to an input as 'Movie', whilst the input in the service ontologies may be referred to as 'Film' – the table of key-value pair IOPEs creates a semantic mapping between the different terms used. The following section describes how abstract matches are used to find concrete matches.

4.4.2.2 The Signature Matching Process

The signature matching process tries to determine if the IOPEs in the service request can be directly mapped onto concrete bindings in the service interface by processing all the service ontologies. SISM processes the Service Profile [DAML 2003c] and retrieves the values associated with each IOPE. These values specify which 'Atomic Process' [DAML 2003c] each IOPE belongs to in the Service Process Model. The IOPEs may have been matched at an abstract level, however they may belong to different atomic processes, therefore SISM needs to determine if a single atomic process exists that supports all the IOPEs in the service request. If an atomic process is found this means that an operation in the service interface exists. In this instance SISM extracts the operation name from the Service Grounding and retrieves the parameter order and the endpoint address from the service interface, which are used to describe how the service is invoked. During this process the table of matched IOPEs are used to bridge between the different terminologies used in the service request and the service ontologies. If SISM maps the IOPEs in the service request to IOPEs in the Service Process Model it tries to determine if the type information supported by both sets of IOPEs match. SISM supports two types of matches at the concrete level: direct matches and indirect matches. These are described below:

Direct Matches: The following tests are performed by SISM to determine if a direct match has been found. If all the tests are true then the service can be invoked without the help of any intermediary services, which is discussed later in this section.

- Test 1: An 'Atomic Process' in the service process model for 'Service A' has associated input elements that conceptually match the inputs described in the service request.
- Test 2: The type information associated with the 'Atomic Process' input 'range' elements for 'Service A' conceptually match the type information for inputs described in the service request.
- Test 3: The 'Atomic Process' in the service process model for 'Service A' has an associated output element that conceptually matches an output described in the service request.
- Test 4: The type information associated with the 'Atomic Process' output 'range' element in the service process model for 'Service A' conceptually matches the type information for the output described in the service request.

A direct match allows the querying device to directly invoke a service without the help of any intermediary services. An indirect match is more complex and is explained below.

Indirect Matches: If a direct match cannot be found, SISM performs the following tests to determine if the service can be invoked using one or more intermediary services.

- Test 1: An 'Atomic Process' in the service process model for 'Service A' has associated input elements that conceptually match the inputs described in the service request.
- Test 2: The type information associated with an 'Atomic Process' input 'range' element for 'Service A' is incompatible with the type information for an input described in the service request.
- Test 3: An intermediary service exists' called 'Service B' that has an 'Atomic Process' input element that conceptually matches the input described in the service request. The type information associated with the 'Atomic Process' input 'range' element conceptually matches the type information for the input described in the service request. 'Service B' has an 'Atomic Process' output 'range' element that conceptually matches the conflicting input described in the 'Atomic Process' for 'Service A'. The type information associated with the 'Atomic Process' output 'range' element in the Atomic Process for 'Service B' conceptually matches the type

information for the conflicting ‘Atomic Process’ input ‘range’ element in the ‘Atomic Process’ for ‘Service A’. This process is recursive and can potentially involve several intermediary services before a solution is found, *i.e.* ‘Service B’ may need to use ‘Service C’ and ‘Service C’ may need to use ‘Service D’.

- Test 4: Anything else fails.

The challenge is to enable devices to form compositions between services either directly or indirectly. For example in Figure 4.13 “DVD Player 1” reads the data from a movie disk the user has inserted into the player. The ‘Player’ service discovers that the media format is Xvid, which it cannot process because it only has a MPEG-2 decoder. If the data format had been MPEG-2 then “DVD Player 1” could have decoded the data using its ‘MPEG-2 Codec’ and transmitted the decoded data to the ‘Visual’ service provided by the Television. However in this instance the data format is Xvid, consequently the SISM Service implemented in “DVD Player 1” tries to resolve the conflict using an intermediary service, which takes as input an Xvid data stream and generates an ‘MPEG-2’ output stream.

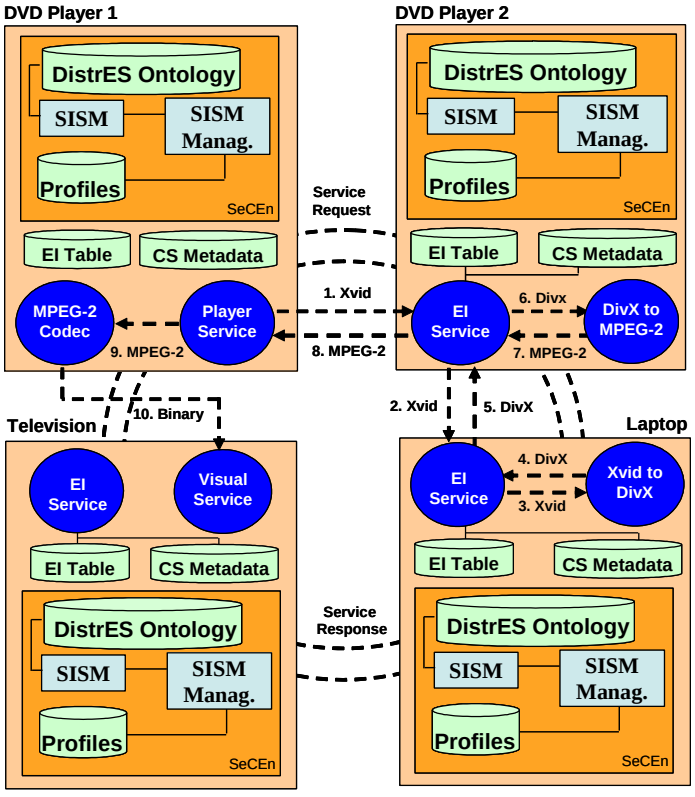

Figure 4.13 Dynamic Service Composition using SISM

Finding an intermediary service is achieved by propagating a reformulated service request to the network describing the IOPE requirements. In our simple example “DVD Player 1” finds “DVD Player 2”, which can indirectly stream an Xvid movie into a ‘MPEG-2’ media format using a service provided by a Laptop. “DVD Player 2” uses the Laptop to convert the Xvid

format into a DivX format, which it can then process and convert into MPEG-2. When this composition is executed, the Xvid data is transcoded and the resulting MPEG-2 stream is decoded by “DVD Player 1” and streamed to the ‘Visual’ service provided by the Television. This allows “DVD Player 1” to extend the interface to the ‘Player’ service it hosts to accommodate the new ‘Xvid’ movie format. “DVD Player 1” is not aware of the composition between “DVD Player 2” and the Laptop and is only concerned that “DVD Player 2” can successfully convert the ‘Xvid’ data into ‘MPEG-2’.

The SISM service achieves this using an Extended Interface Metadata Object (EIMO). The EIMO describes how signatures are constructed to transcode data and indicates whether the intermediary service itself can be directly invoked or whether it also requires intermediary services. This process allows services to dynamically discover and resolve IO conflicts that may occur and proactively establish compositions with intermediary services.

When intermediary services are discovered this may result in several candidate services that provide the same functionality. In this instance the services that best match the device capability requirements as described in section 4.3.2 on page 88, which are defined in the service request, will be added to the EIMO.

In the following section we describe how the EIMO is invoked using the Extended Interface Service.

4.4.2.3 The Extended Interface (EI) service

The EI service, as illustrated in Figure 4.14, is invoked when a service provided by the device does not directly support a method invocation.

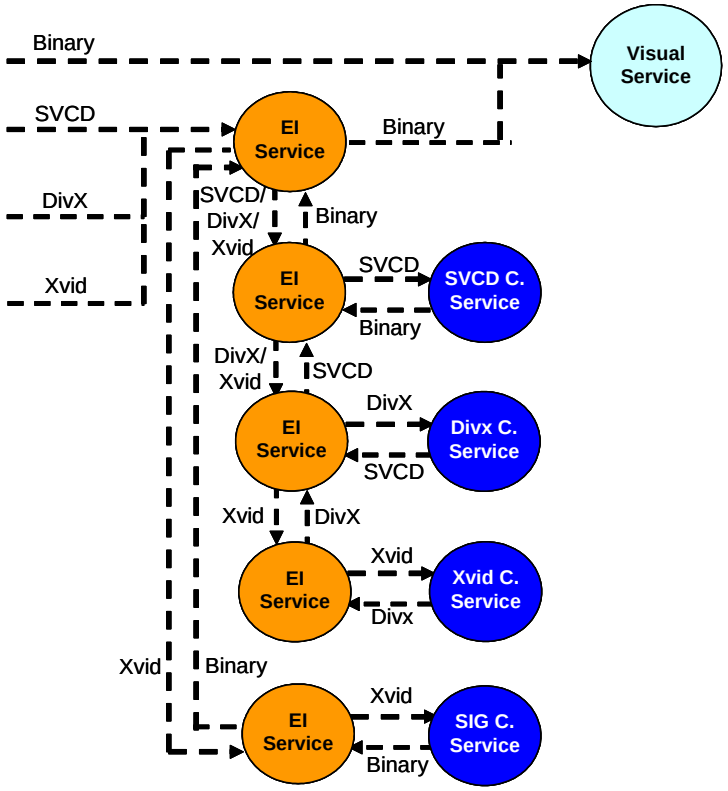

Figure 4.14 Extended Interfaces for the Visual Service

This service has a fixed operation name called 'EI' which takes two parameters – the first parameter is the EIMO and the second parameter is an object array which contains all the parameters required to invoke the intermediary service. The EI service processes the EIMO, which provides information about the operation name for the intermediary service, the parameters it takes, including the associated data type information, and the order in which the parameters appear in the signature.

The EIMO also specifies the connection mode supported by the service. If the connection mode is 'direct', the EI service uses the metadata for the intermediary service to construct the required signature using the parameters in the object array, before binding with it and executing the required method. In this instance 'direct' means that device *A* can directly use a service S_1 provided by device *B* without having to use any intermediary services. If the connection mode is 'composite' then the EI service processes the EIMO for the intermediary service it needs to use before connecting to its EI service and passing it the metadata and the parameters. In this instance 'composite' means that device *A* indirectly uses a service S_1 provided by device *B* via service S_2 provided by device *C*. This process continues until a direct connection with a service in the composition is made.

This mechanism ensures that the service interface evolves over time to accommodate numerous other inputs it was not initially designed to process. For example a DVD Player that only implements an MPEG-2 codec can read a number of different media formats and interact

with the ‘Visual’ service by first transcoding the data it reads from the disk into binary data by discovering and using data adaptation services. The following section describes how the SISM Service has been designed.

4.4.3 The SISM Service Design

The following models describe how service requests are processed and how abstract and concrete matches are performed. They also describe how signatures are built, how intermediary services are discovered and how peer services are invoked.

Figure 4.15 illustrates how service requests are processed by the SISM service. The service request is matched at an abstract level and if a match is found the service ontologies for the service, including the service interface are retrieved and passed to the concrete matching algorithm. This algorithm uses the service ontologies and tries to map the semantic metadata to concrete signatures contained within the service interface. If a match is found the service advertisement for the current service being processed is returned to the service requester. If a match cannot be found the matching process fails and a null value is returned.

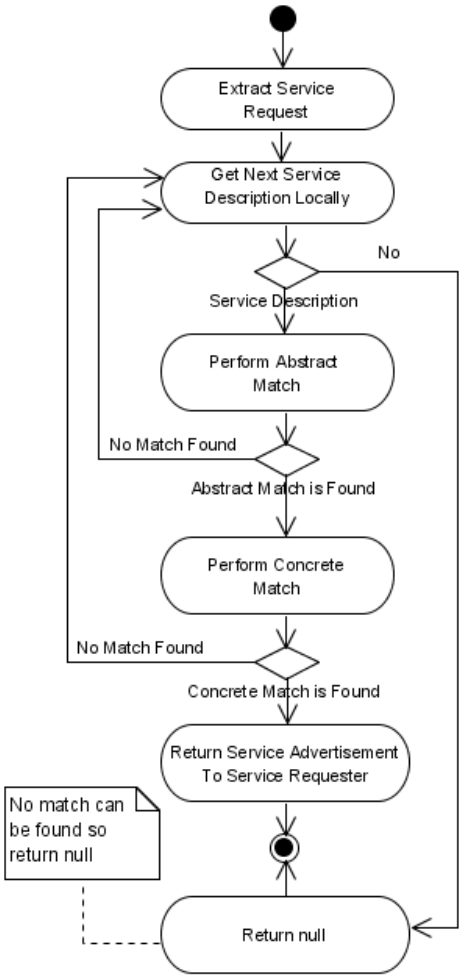

Figure 4.15 Process Service Request

The abstract matching process itself begins by iterating through the IOPEs described in the service request and extracting each IOPE in turn. IOPEs are used in the Service Profile to capture the *inputs* and *outputs* service signatures support within the Service Interface including any *preconditions* that must be satisfied and any *effects* that result when the service is executed. Figure 4.16 illustrates the class diagram for the IOPEs which describe the class variables used and the methods supported.

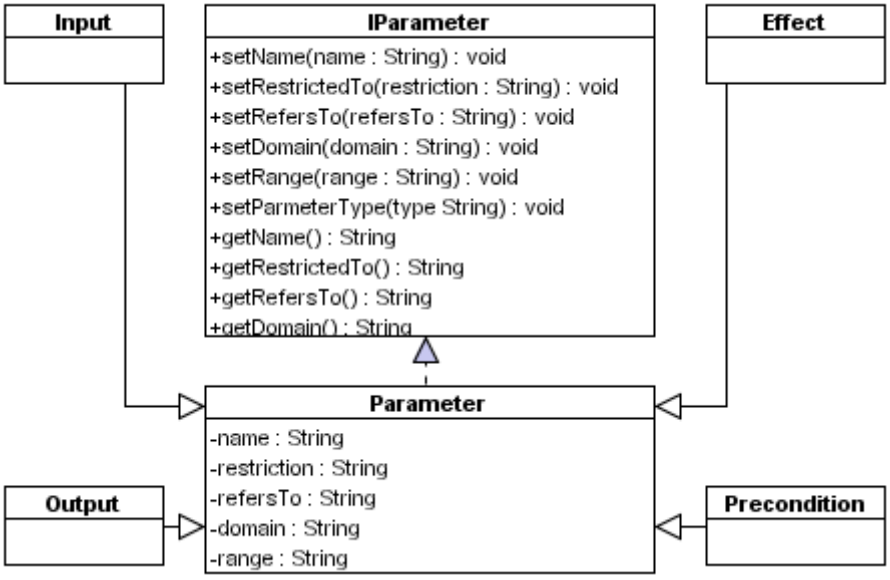

Figure 4.16 IOPE Class Diagram

The IOPE in the outer loop (service request) is compared with each IOPE extracted in the inner loop (service description) to determine if an exact match can be found – this being the case the service request IOPE status is set to true. If an exact match cannot be determined this process calls the semantic interoperability process to determine whether any ontologies within the wider network have a relationship that links the two terms together semantically. If a semantic relationship is found then the service request IOPE status is set to true, again indicating a match has been found via some semantic relationship. When all the IOPEs in the service request have been processed, this process checks to see if all the service request IOPEs statuses are set to true. If this is the case then an abstract match has been found. This process is illustrated in Figure 4.17.

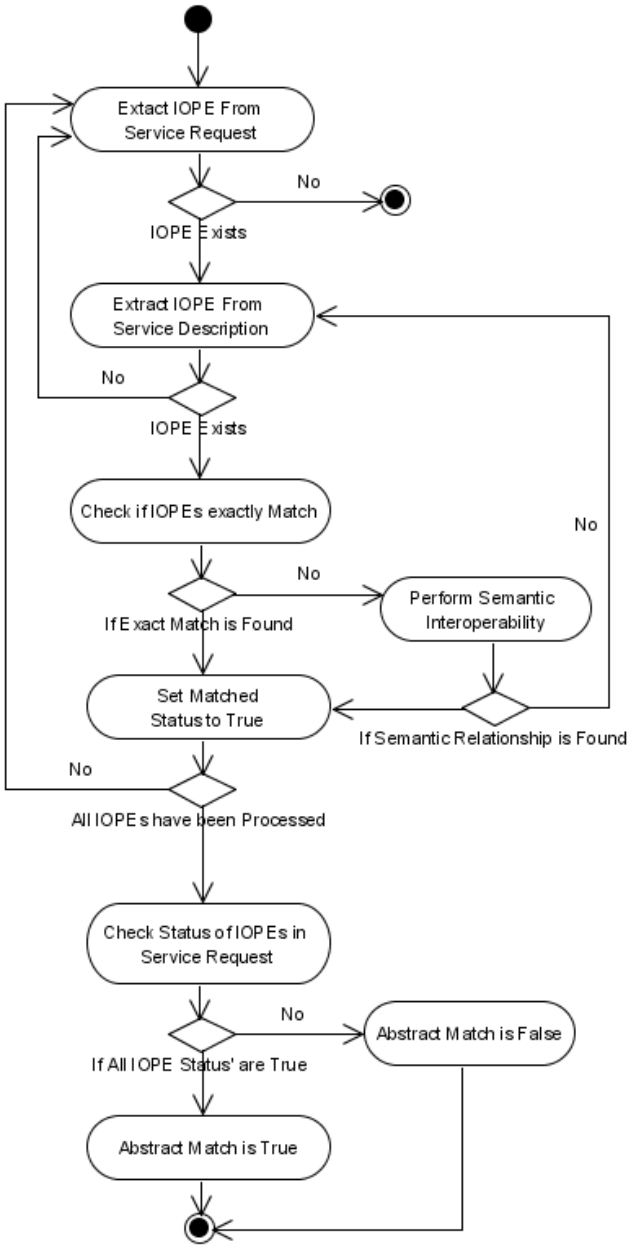

Figure 4.17 Perform Abstract Match

If an abstract match is found the service ontologies including the service interface are passed to the concrete matching algorithm. This process begins by iterating through the IOPEs (*inputs*, *outputs*, *preconditions* and *effects*) in the Service Profile Model and tries to find a corresponding *Atomic Process* in the Service Process Model as illustrated in Figure 4.18. Atomic Processes are used in the Service Process Model to logically group *inputs*, *outputs*, *preconditions* and *effects* to form abstract semantic signatures. This is a key technique used to map high-level semantics to low-level service interfaces. When all the IOPEs in the Service Profile have been processed a check is made to determine whether a single *Atomic Process* exists in the Service Process Model that subsumes all the IOPEs contained in the Service Profile.

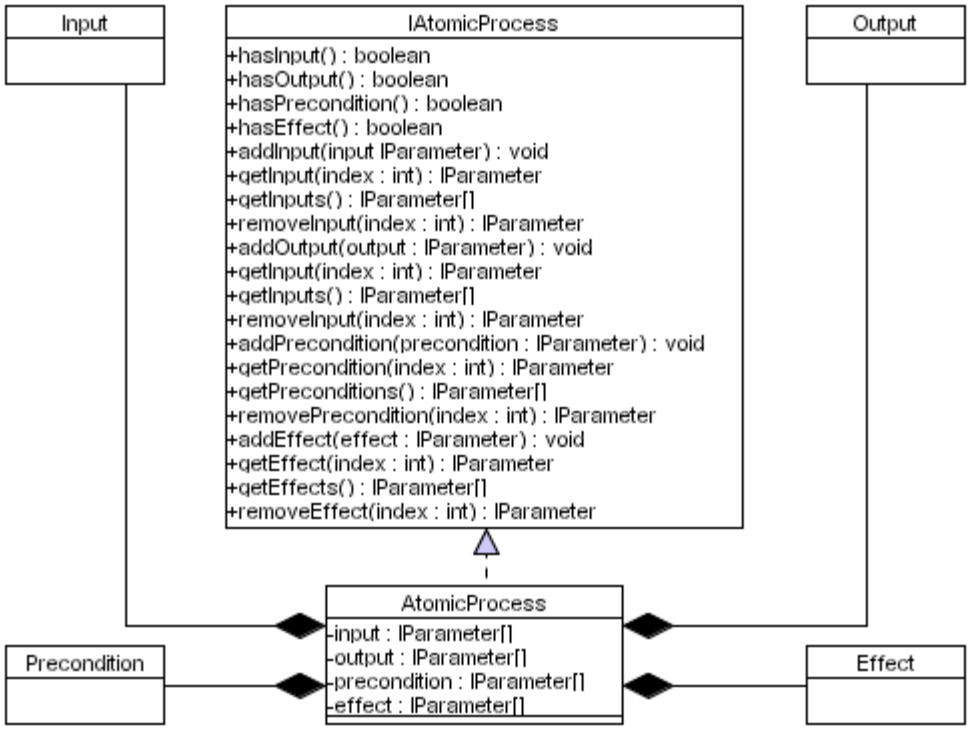

Figure 4.18 Atomic Process

If an *Atomic Process* is found the Build Signature process is called to determine if the *Atomic Process* can be mapped onto a concrete signature described in the Service Interface. If the Build Signature process returns a Service Advertisement then this constitutes a concrete match and the advertisement is returned to the service requester. This process is illustrated in Figure 4.19.

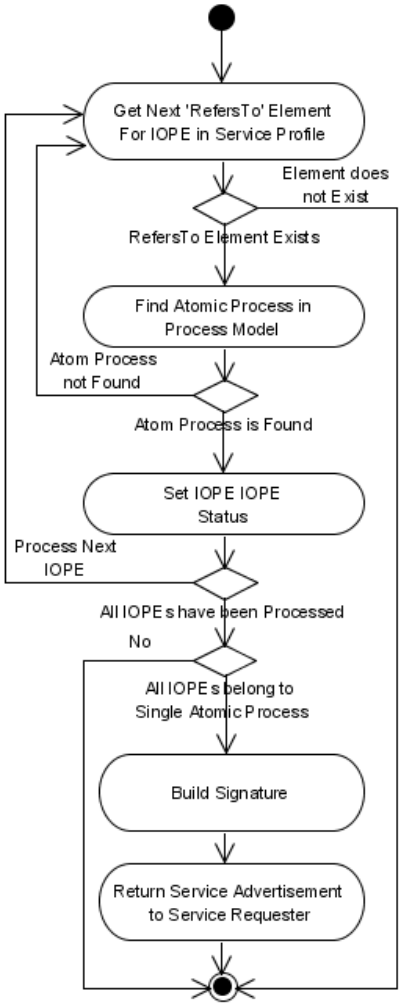

Figure 4.19 Perform Concrete Match

An important design requirement within this framework is to resolve matching conflicts that occur. If the Concrete Matching Algorithm successfully maps the IOPEs in the service request with signatures in the Service Interface but encounters data type conflicts it must try to resolve these conflicts using intermediary services that can explicitly provide data type conversions. Contrary to this requirement there may be instances when converting the data type is insufficient and rather the software to achieve this must be downloaded. For example a device may choose to download a particular codec, from an intermediary service, to process some media format rather than converting the data stream in real-time to increase or maintain a high quality of service. However less capable devices may choose to outsource the processing to a more capable device. How intermediary services are discovered to resolve IOPE conflicts is discussed later in this section.

The build signature process illustrated in Figure 4.22 is the core component within the concrete matching algorithm and is used to determine if a concrete match is found. If the signature can be built this means that the IOPEs including their data type information can be

mapped onto a signature contained in the Service Interface. The build signature process begins by trying to find an *Atomic Process* in the Service Grounding Model using the *Atomic Process* extracted from the Service Process Model. The Service Grounding describes how the service can be accessed and controlled. The grounding contains one or more *AtomicProcessGroundings*, which contain Service Input objects, Service Output objects, Service Operation objects, Service Input Message objects and Service Output Message objects. These objects allow the IOPEs in the Atomic Process Grounding object to be mapped onto concrete signature bindings in the Service Interface. The class diagram for the Service Grounding Model is illustrated in Figure 4.20.

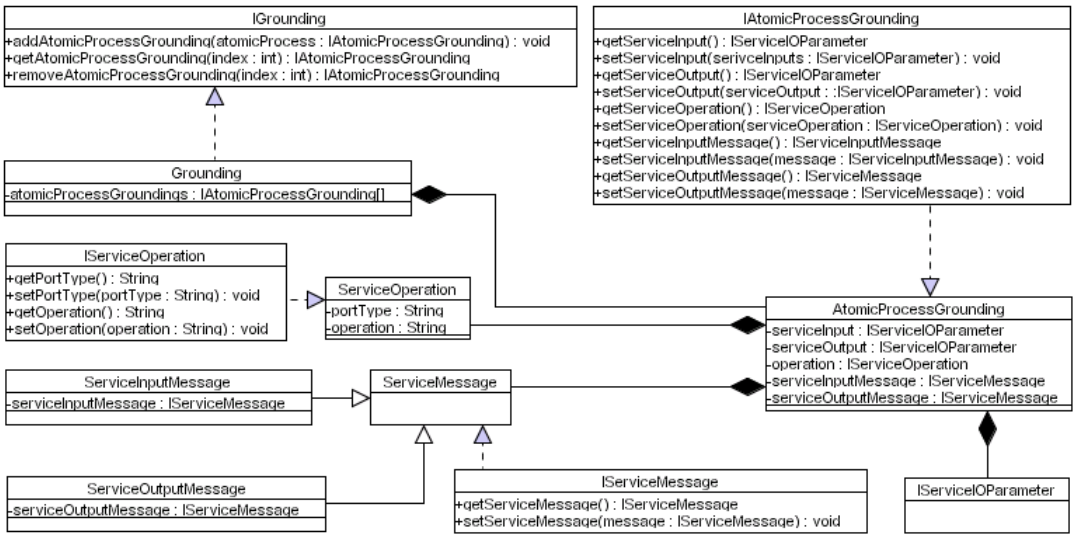

Figure 4.20 Service Grounding Model

If an *Atomic Process* is found the operation name associated with the *Grounding Atomic Process* is extracted, otherwise it is terminated. The operation name may be part of the Service Grounding *Atomic Process* or it may be extracted from the Service Interface. The Service Interface describes the signatures a particular service supports. Each signature is known as a binding and a Service Interface may support several bindings. Each binding contains a *portType*, which contains an operation. An operation contains input message objects that describe all the inputs the signature supports and an output message object that describes the output the signature returns (if a return value is used). Each message contains the name of the message and one or more message parts. Each message part contains the *partName*, the *parameter* and *parameterType*. The class diagram for the Service Interface is illustrated in Figure 4.21.

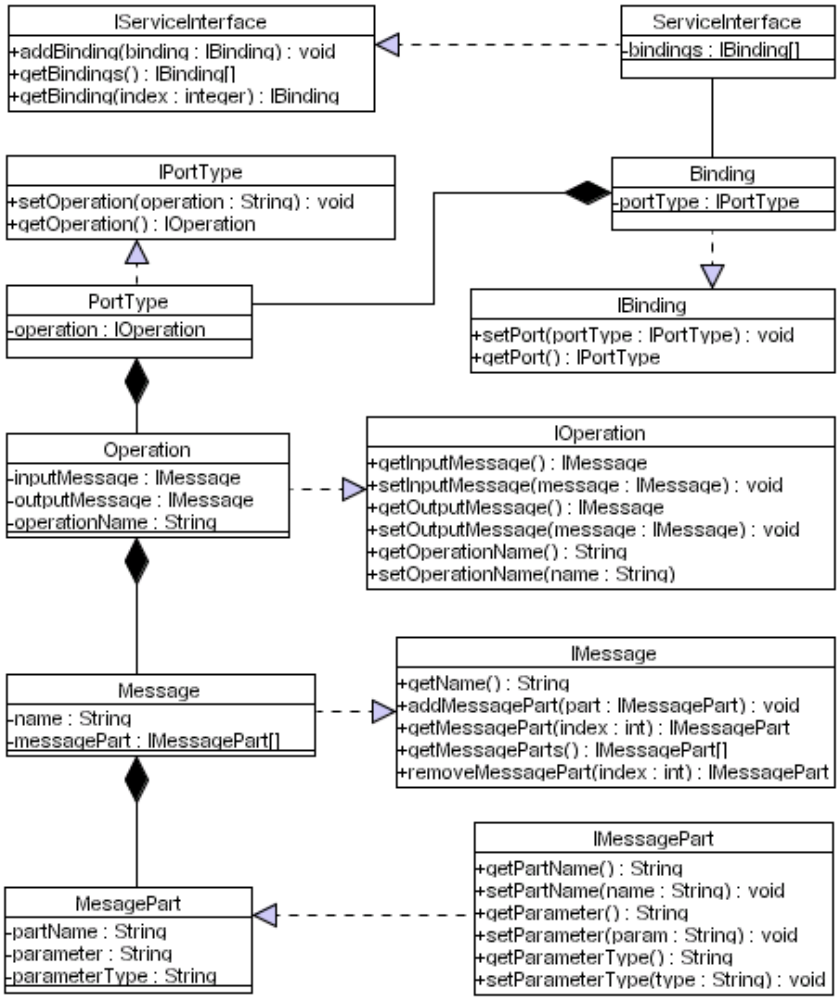

Figure 4.21 Service Interface Model

Once the operation name has been extracted this process iterates through the IOPEs in the service request and extracts each IOPE in turn. When an IOPE has been extracted it is used to extract the corresponding IOPE in the Service Grounding *Atomic Process*. Once these two IOPEs have been extracted the data type information for each IOPE is matched and if a match is found the IOPE data type status in the service request is set to true. If a data type conflict is discovered then the conflict is resolved using an intermediary service – if the conflict is resolved the IOPE data type status in the service request is set to true, otherwise the next IOPE in the service request is extracted. When all the IOPEs in the Service Process Model have been processed, this process checks to see if all the IOPEs, including there corresponding data type information, have been matched – in this instance all the data type status values should be true. If this is the case then the service endpoint is extracted from the Service Interface and the Signature Built status is set to true to indicate a concrete match has been found. In this instance the metadata is returned to the service requester. If any status values are false then no concrete match has been found and this process terminates. This does

not mean that a service does not exist, it just means that the device processing the request cannot provide the service. In reality numerous devices will process the service request, as such it is envisaged that in a large P2P network as least one device will be able to satisfy the request. This process is illustrated in Figure 4.22.

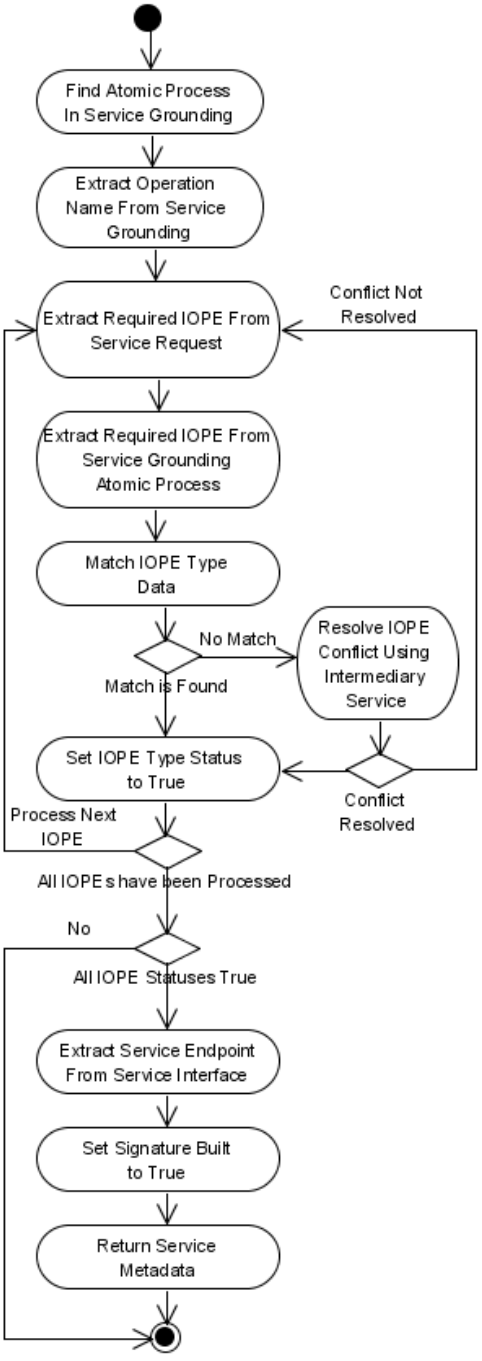

Figure 4.22 Build Signature

One of the important requirements within this research is to develop a mechanism that allows conflicts within signatures to be resolved using intermediary services. This means that conflicting parameters are converted into the expected data type using a service discovered

within the network. This is achieved by extracting the conflicting IOPE name and data type from the signature, including the required IOPE name and data type and inserting them into a newly created service request. This service request is used to discover an intermediary service that can convert the conflicting data type into the required data type. If a service cannot be found then this process terminates. If a service is found an Extended Interface file is created and the extracted IOPEs are added. The service advertisement for the intermediary service being used to resolve conflicts is also added and once this has been done the Extended Interface file is returned to the service requester. This process is illustrated in Figure 4.23.

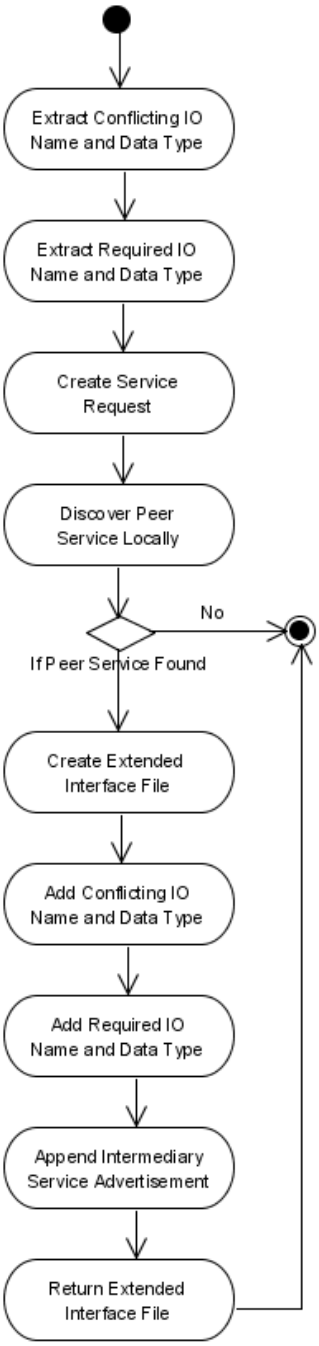

Figure 4.23 Find Intermediary Service

When Extended Interface Objects are returned to the service requester, they are used to determine how the service can be connected to and invoked. First the required Extended Interface file is retrieved and the service advertisement is extracted. Once the service advertisement has been extracted, the service invocation mode is extracted and used to determine whether the service can be directly invoked or invoked via an intermediary service. If the connection mode is *direct* then the *direct* endpoint is extracted else the *composite* endpoint is extracted. The endpoint is used to bind to the service. If a connection mode cannot be established then this process terminates. If a *direct* connection is established with the service then the required signature is built by extracting the method name, the required parameter names, including the data type information, and the order in which the parameters must appear in the signature. This information is then used to build the signature and invoke the required method. If a *composite* connection is established with the service, the service advertisement along with the parameters is sent to the *composite* endpoint for further processing. This involves extracting the information regarding the intermediary service being used, again as illustrated above, the connection mode is determined and the intermediary service is either, directly bound to and invoked, or the metadata is sent to the intermediary service. When the intermediary service is invoked the conflicting data is substituted with the value the service has returned. This process is illustrated in Figure 4.24.

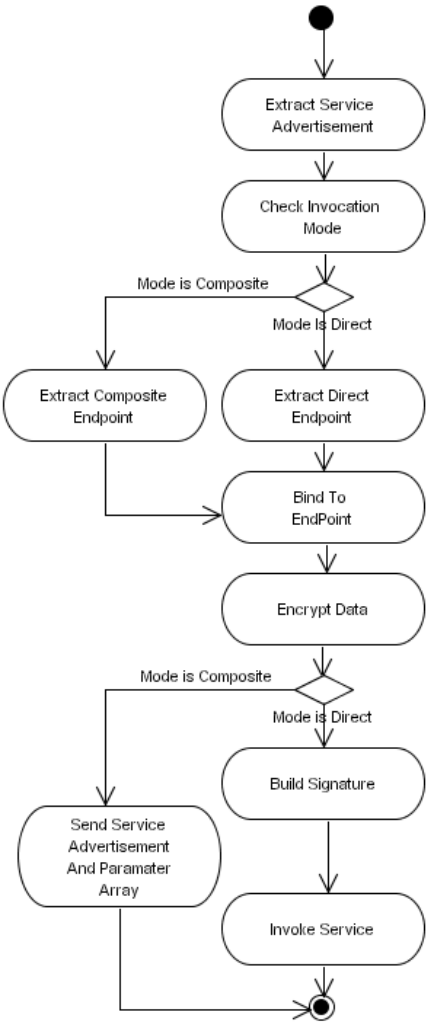

Figure 4.24 Invoke Peer Service

For a full list of UML diagrams for all the secondary services that comprise NASUF see Appendix A, B, and C.

4.5 Summary

This Chapter describes the high-level design requirements considered within this thesis for the secondary services that comprise the NASUF architecture (detailed UML models can be found in Appendix A, B, and C). It describes how services are semantically described in terms of their capabilities using ontological structures and dynamically composed to extend devices beyond what they where initially designed to do. This included a detailed discussion regarding how semantic interoperability is addressed between the inherent terminology differences used by different device manufacturers. Devices continually evolve local ontology structures to reflect these changes, ensuring that devices learn and map the terminology they use with terminology used by other devices based on general consensus.

This chapter also argued that semantic descriptions themselves help solve the interface problem and the design decisions used within our framework illustrate that this approach can form compositions between other services based on the capabilities services describe using semantic metadata, without having to know the concrete interface bindings beforehand.

Devices support different capabilities and as such some devices will be better equipped to provide a given service than others. Consequently, the design considerations illustrate that services are selected based on how effectively the device can execute the service.

Device configurations are automatically managed using the framework services, which provide self-adaptation mechanisms that detect and make compensatory changes when environmental changes are detected. This abstracts the underlying complexity associated with device composition and network configuration, from the user, which is a feature not present in existing approaches discussed in Chapter 2.

Chapter 3 and Chapter 4 provide formal design models that describe how our framework addresses the requirements, and overcomes the challenges, described in Chapter 1. High-level use cases, algorithms and data models illustrate how each service operates and what data structures are used. Our design puts forward a viable solution that goes far beyond current solutions such as OSGi, UPnP and DLNA.

It is hard to show how one set of services alone (presented in Chapter 3 and this chapter) could provide a consumer much benefit, however when coupled together they provide value-added functions that surpass existing middleware architectures described in Chapter 2. There is a coupling (dependency) in our services but this is enabled via P2P which makes for a more robust and redundant latent capacity within the entire network. Again this is a feature not present in solutions such as OSGi.

Our design shows how machine-processable semantics can be used to overcome the inherent limitations associated with attribute-value pair matching which is a technique used in OSGi, UPnP and DLNA. Furthermore the design formalises how devices can be composed and self-adapted to make compensatory changes to device configurations based on environmental changes.

In the next chapter we discus a case study used to demonstrate the services provided by our framework, which is used as a bases for our implementation discussed in Chapter 6.

Chapter 5

5 Case Study: Intelligent Home Environment

5.1 Introduction

In this chapter a case study is presented that is used to demonstrate the functions provided by our framework, which has been implemented as a prototype – this is discussed in more detail in Chapter 6. The case study describes an intelligent home environment capable of seamlessly integrating networked appliances, such as TVs, Media Players, Surround Sound Speakers, and Hifi Systems. It illustrates how our framework can be used to dynamically compose services provided by these devices such as Visual, Audio and Player services. The successes and failures found during the development of the case study are also presented as well as other possible application areas that our framework can be used for.

5.2 Case Study

In this section an intelligent home environment is proposed that allows networked appliances to automatically interconnect and dynamically form relationships with devices connected to the network. This system allows devices to self-adapt and continually provide the best quality of service based on devices and services within the current environment. If devices or services fail, alternative solutions are automatically composed without any human intervention, with minimal disruption to the user. The case study developed automatically interconnects the audio/video and player devices within a typical home environment. Each device publishes the functions it provides as independent services. For example a TV publishes the visual, audio and RF-Receiver functions as independent services that can be simultaneously discovered and used within the home environment.

This case study was selected to: (i) test the design decisions presented within this thesis and illustrate how an Intelligent Home Environment can be created that addresses several limitations found with traditional home networking solutions; (ii) demonstrate how devices and services can be utilised by publishing and using functions provided by devices simultaneously without disrupting devices and services currently in use; and (iii) highlight the flexibility associated with the service-oriented architecture used in NASUF which allows devices irrespective of their capabilities to be interconnected.

Imagine your home environment, more specifically your living room, and the devices it contains. It is more than likely that it has a DVD player, VCR, Widescreen or Plasma TV, a surround sound speaker system, and a HiFi. Now imagine the time you bought your DVD Player and tried to integrate it with you existing device configuration. Like most people, you may have taken the DVD player out of the box and attempted to connect the wires to your TV and surround sound system and one hour later decided that maybe you need to look at the instructions. After a further hour trying to understand the instructions, tune in your TV and configure your surround sound system you finally succeeded in viewing the DVD movie you bought.

These kinds of experiences are becoming increasingly more common because devices and their associated configurations are becoming more complex, thus requiring considerable effort from specialists and home users alike. This is set to become more difficult as the growth of personal computer usage, the Internet and networked appliances become more widely used in more diverse applications than ever before. We can expect ordinary everyday appliances to become part of these networks, and networked devices will become pervasive and often invisible to the users.

Now imagine a future environment whereby you take the DVD player out of the box, switch it on and it just works. You put your DVD movie into the player, press play and the video is displayed on your Plasma TV and the sound is streamed to your surround sound speaker system. No manual configuration was required to integrate the DVD player and you did not have to tune in your TV or configure your surround sound system. When the DVD player is switched on it automatically communicates with all the other devices within the home via its wireless network interface. These devices automatically form relationships with other devices in the home based on what data the device outputs and what inputs devices process. This is analogous to a jigsaw puzzle whereby the shapes of the individual pieces act as interfaces that can be directly composed with corresponding interfaces provided by other jigsaw pieces.

Taking this vision one step further, devices will be highly flexible and will encompass mechanisms that allow them to self-adapt based on conflicts during the integration process or changes within compositions. In the former case devices will not simply fail but rather proactively attempt to rectify the problem. Returning to our DVD example, imagine if you put a movie into the player, which is encoded in a format your player does not have a codec for. In this instance the DVD player could do one of two things. It could automatically discover an intermediary device capable of processing the unknown movie format, which transcodes the data into a format the DVD player is able to process. Alternatively the player could automatically locate the codec internally within the home network or via the Internet, download it and use it to play the movie. Making devices network-enabled in this way opens

up a number of possibilities that will not only become more important in the future, but which will allow devices to be proactive.

Mechanisms will also allow devices to sense its own internal changes including changes amongst devices it has direct relationships with. Again returning to our DVD example, if the player determines that the surround sound system has become unavailable for some reason, this change will be sensed and the player will automatically try to discover an alternative set of speakers capable of processing the audio stream. In this instance the player could use the speakers provided by the Plasma TV screen or the speakers provided by the HiFi and continue streaming the audio with minimal disruption to the user's viewing experience. If the surround sound system becomes available again the player will again sense this change and determine that the surround sound speakers provide a better multimedia experience and as such stop streaming the audio to the Plasma TV speakers and begin streaming the data to the surround sound system.

The Intelligent Home Environment has the provision to provide any number of visual, audio and player services. Once devices have been switched on, they all form relationships with each other based on what devices want and what devices provide. For example the audio and visual services offered by a TV appliance could be combined with the player service offered by a DVD appliance to form a 'Home Theatre System'. Alternatively, the audio service offered by a Hi-Fi appliance could be combined with the visual service offered by a TV appliance and the player service offered by a DVD appliance. This is defined as Function Utilisation and is illustrated in Figure 5.1.

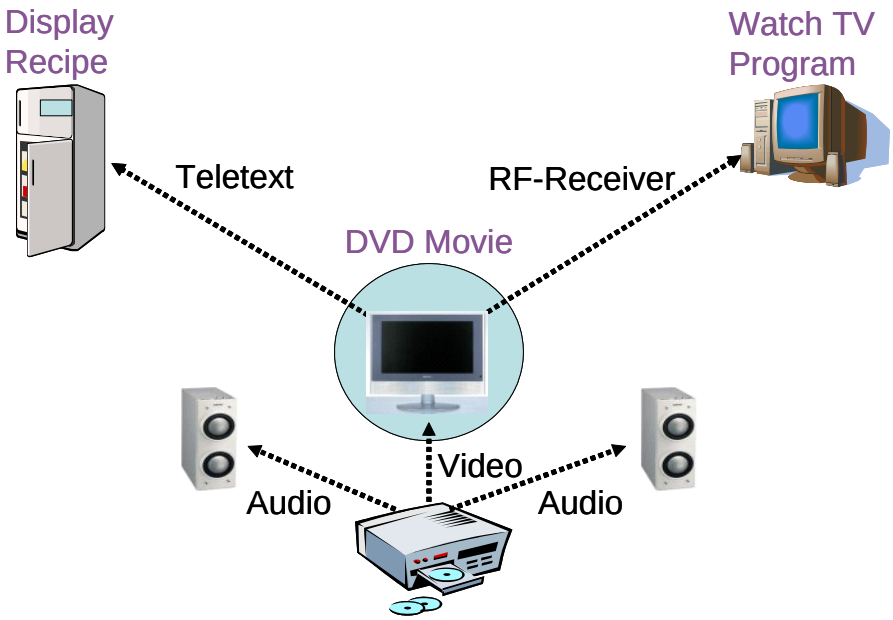

Figure 5.1 Function Utilisation

This provides additional advantages to the home environment, which enables devices and services to be composed to create applications that do not explicitly need to be installed, but rather can emerge based on device composition. The emergent functionality created is dependent on what devices exist within the environment and the services they provide at any given time. One example of an emergent function may be a virtual intercom system, which is comprised of all the available speakers within the environment and a microphone provided by a mobile phone as illustrated in Figure 5.2 Virtual Appliance

.

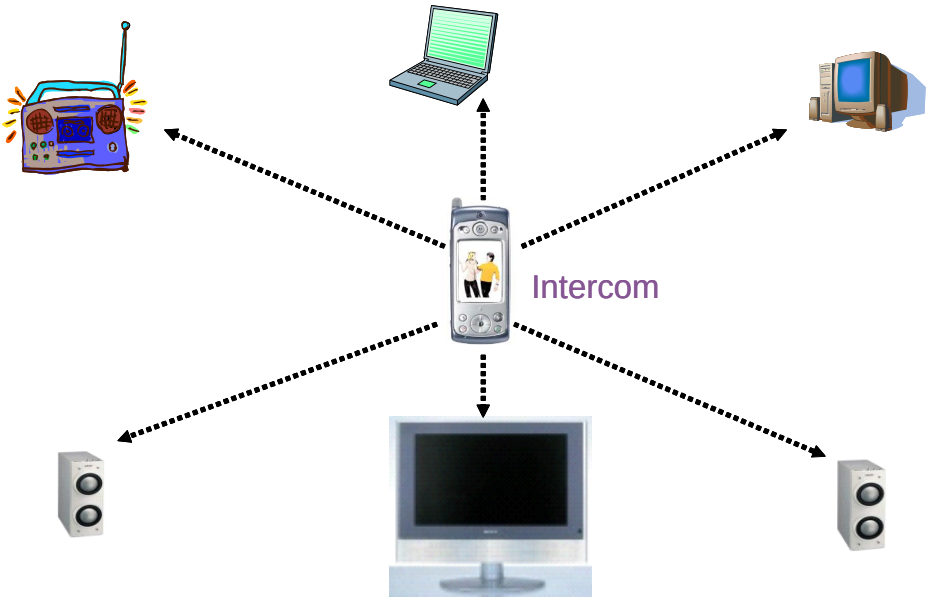

Figure 5.2 Virtual Appliance

In this instance the intercom system does not explicitly exist, but rather emerges when devices are composed. The NASUF middleware ensures that devices are not carefully manufactured, but rather are an emergent property directly attributed to how devices are connected within that environment and the functions they support. How devices are used and composed at higher levels is application specific and is dependent on the application requirements, which when executed are controlled for the duration of the task and then released. Consequently solutions are not bespoke and compositions are not dependent on pre-determined configuration rules. The integration process is based on how well the capabilities provided by devices map onto the user requirements for the task in hand. Depending on the application domain, networked devices are combined in any number of ways to perform some function.

Demonstrating the self-adaptive nature of NASUF devices can automatically select alternative devices or services that provide a better quality of service. One possible example as illustrated in Figure 5.3 is the redirection of audio and video from a video-enabled mobile phone.

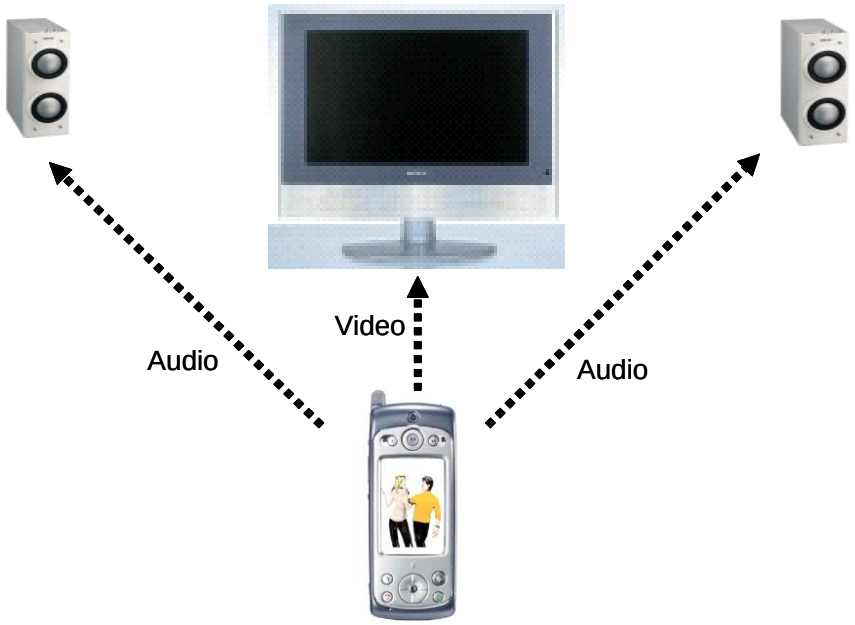

Figure 5.3 Dynamic Service Composition

During a video call you enter your home environment and your phone automatically integrates itself within the network and discovers the devices and services it has relationships with. In this instance the phone discovers a visual service provided by a television and an audio service provided by a surround sound speaker system. Based on the capabilities of the mobile phone and the newly discovered devices, the phone can automatically self-adapt and redirect the video and audio content to the more capable devices. The user still uses the microphone provided by the phone except the video is displayed on the TV and the audio is processed by the surround sound speakers.

NASUF provides the flexibility to combine any of the services available into a specified configuration to form device compositions within the home. The composition process itself is based on device capability matching, so although many devices form relationships based on the behaviours they support, active compositions are constructed based on the overall quality of service devices provide. Initially a composition may consist of a DVD player, a surround sound speaker system and a 48inch Plasma screen, which the middleware has composed to give the user the best viewing experience. However one of the features offered by NASUF is that in the event of one of the devices becoming unavailable, for example the surround sound speaker system, it can automatically adapt and select alternative speakers, *i.e.* speakers offered by the Plasma screen, to process the audio stream. Furthermore the middleware can revert back to a previous configuration if and when better services come back on line or are newly installed. So for example, if the surround sound speaker system comes back online the current audio service is stopped and the surround system is selected as the best solution and started.

5.2.1 Characteristics of this study

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

a) Devices join the network and automatically form compositions with other devices within the network.
b) Devices can be used to perform some composite function. For example when the DVD player's play button is pressed the player automatically selects and connects to the best audio/visual services it is aware of.
c) Devices are selected that provide the best quality of service based on what devices and services are available within the home network.
d) Device and service compositions can automatically self-adapt in the advent of device or service failure by selecting the next best service, connecting to the device that provides it and continue the composite execution.
e) Services provided by devices can be used in conjunction with other services being used without affecting current service compositions. For example if the visual service provided by the TV is being used to watch a DVD movie, the RF-Receiver can be simultaneously used to display a terrestrial TV channel on the PC located elsewhere in the home, without disrupting the persons viewing experience.
f) Virtual appliances can be automatically discovered and composed to create applications that have not explicitly been installed. For example the microphone provided by a mobile phone could be used to broadcast a message throughout the home by using all the available audio services. This results in a virtual intercom system that has not been explicitly installed.

These characteristics demonstrate how an intelligent home environment can be used which utilises the available operational functions provided by devices; creates virtual appliances and dynamically composes devices and services to create some high-level value added function not provided by one single device or service alone.

5.2.2 Using our Framework for an Intelligent Home Environment

Several steps need to be taken to configure NASUF to implement the Intelligent Home Environment. These are described within this section.

Step 1: Creating the Device objects – in this case study Audio, Video, Player and Controller objects are created and are implemented on multiple machines within the experimental environment, which is discussed in more detail in Chapter 6. These device objects implement the secondary services that comprise NASUF, which may be explicitly implemented on the device itself or used remotely within the network. The Controller device is a special device

used to discover and control devices and services within the network. Using the Controller, devices can be stopped, started and invoked. The Controller also allows services provided by a device to be stopped and started. When devices are discovered, the associated devices and services used by that device are also displayed, which can also be controlled. When a device is executed, the composite services it uses are automatically controlled via devices that use these services.

Step 2: Creating NASUF Secondary Services – Depending on the device's capabilities the secondary services are explicitly implemented on the device. In the case study each device implements the DeCap, DistrES and SISM services. Although devices such as audio speakers may not be capable of implementing all these services in a real-world setting they have been implemented to evaluate how devices function when secondary services are added and removed. The idea is that even if only one device provides the secondary services they can be shared and used by all other devices within the network, however overall performance will decrease because multiple devices are trying to use the same secondary services.

Step 3: Creating the Application Specific Peer Services – Device objects implement application peer services which expose the device's functions. The Audio and Video devices use a Multimedia Receiver peer service configured to either receive audio or video streams dependent on the device implementing the service. The Player device uses a Multimedia Transmitter peer service configured to transmit audio and video multimedia streams.

Step 4: Starting Devices – When the Audio, Video, Player and Controller devices have been created and their associated secondary and application specific peer services started, the device itself is started. At this point the device and the services it provides can be used by the device and any other device within the network.

Once these steps have been completed a combination of devices and services can be combined to provide high level functions. For example the Player device can combine one or more of the Audio and Video devices to create a Home Theatre System. Compositions are constrained based on the semantic queries propagated within the network and the semantic descriptions used to describe services. In this instance Video devices will not form compositions with Audio devices because they do not share any functional relationships. Both devices process multimedia streams, consequently these devices receive input but do not provide output. Typically compositions are formed based on what data devices *output* and what *inputs* they receive, including any *preconditions* and *effects* that need to be considered.

In a typical home environment multiple services of the same type will co-exist. For example the 43inch TV located in the living room and the 3G mobile phone you have will both provide a visual service. Consequently compositions take into account devices that will provide the

best quality of service. For example the Player device will discover and use the 43inch TV rather than the 3G mobile phone to watch a movie because it will provide a better quality of service. However in the event that the 43inch TV becomes unavailable for some reason, alternative TV visual services will be automatically selected, with the 3G mobile phone being one possible choice. In the case study this functionality is achievable using the NASUF framework.

Using the Controller device the user can discover any device or service within the network. Although individual control can be placed on devices and services, base compositions will already be in place. This is performed when devices are initially switched on. As described above devices automatically determine which devices and services they have relationships with. Using the Control device the user can execute compositions and individually change services within the composition. If device and service failures occur the Control device is automatically updated to reflect these changes. This case also applies to devices and services that re-register themselves within the network.

5.2.3 Anomalies in this Case Study

The service interface file used to describe the signatures the service supports is attached to the service advertisements however only the operation names are extracted whilst the parameters operations supported are disregarded. In this instance operation names such as “Play”, “Listen”, and “Stop” have been used, which typically do not contain any parameters. Discovering and more accurately matching services that contain parameters is the focus of future work.

5.2.4 Positive aspects of this Case Study

This case study provides a number of advantages over other home network solutions. Devices can be automatically deployed and composed without any human intervention. This case study illustrates how zero-configuration can be realised using the secondary services provided by NASUF. Many home middleware architectures are human centric and rely on human expertise to glue devices and services together. In NASUF this process has been automated and devices form loosely coupled relationships between each other based on device capability and peer service capability matching techniques. Typically it is the user that decides what devices to use in order to provide the best composition possible. This is not the case in NASUF, which is capable of automatically determining what devices to use dependent on the services they provide and how effective they can execute those services.

The case study illustrates how device configurations can automatically self-adapt in the event of device or service failure. Using NASUF the home environment continually tries to

interconnect devices and create solutions that provide the best quality of service. In this instance no matter how bad the solution is NASUF will always produce a solution that allows devices to be composed. The self-adaptive nature of NASUF provides additional benefits to home networking solutions that surpass current middleware standards such as OSGi and UPnP. A description of the case study implementation is discussed in more detail in Chapter 6

5.3 Other Application Domains

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

5.3.1 Emergency Installations - Ad-Hoc Integration and Service Utilisation

Emergency installations (fire, ambulance, police and rescue services) are becoming more ad hoc in nature and are adopting technologies that lend themselves to fast moving intercommunications where the topological structure is continually changing shape as and when devices and services are present. As such our framework allows the following requirements to be realised.

- NASUF can provide an 'intelligent' middleware that allows devices and services to be dynamically integrated. As emergency installations move through the environment the network is maintained and automatically adapted as new devices and services arrive and existing devices and services disconnect from the network.
- Independent emergency installations (ad hoc networks), can automatically join and leave other sub-emergency installations as and when different sections occupy the same location, to form one single network, *i.e.* the fire, ambulance, police and rescue services can form a network and share services at an accident scene. This allows services and information within this single network to be shared – when an emergency installation re-locates it takes its devices, services and information with it. This allows for automatic network configuration, information transfer, and device and service utilisation.
- No maintenance or pre-configuration of networks, devices or services is required. The ad hoc nature of decentralised networks ensures that devices within a particular location are automatically interconnected into one logical network. Whilst the

middleware discovers and composes services/functions provided by devices depending on particular functions requested.

5.3.2 Medical Installations – Emergent Functionality

Medical installations such as hospitals require a considerable amount of equipment, as is the case of intensive care units. Such equipment is costly and in most situations the total functionality provided by all devices remains largely redundant because only parts of the functions provided by a device are used. As such costs can be reduced and equipment requirements can be minimised by utilising functions more efficiently. Devices that are typically bought can be created by combining existing functions within the hospital environment, which can be defined as emergent functions. One example could be an observation system used to monitor the patient's heart, temperature, and blood pressure. Instead of having an appliance located within the patients' room small wireless sensors, which implement NASUF, could be used to send data to monitoring services provided by devices located elsewhere in the hospital [Fergus 2004]. The data received could be streamed to a dumb visual display located within the patient's room, however all processing is performed by devices designed to process the data received from the patient.

Technological advances are moving at a fast pace and as such constant upgrades to the existing equipment owned are required. In these instances only small changes are required such as new networking interfaces or media codecs, whilst the core functionality remains the same. For example a device may exist within some installation capable of processing multimedia content in a particular format because it has the required codec. However if a new device is integrated into the environment that uses a different multimedia encoding then this content cannot be processed by legacy devices, consequently requiring a device upgrade. Instead of replacing the device a better alternative would be to allow the device to extend the functions if provides beyond what it was designed to do. When a conflict is encountered, *i.e*. a multimedia format it does not have a codec for, it can either discover the codec within the network, download it and process the content or it could find an intermediary service provided by some device that can transcode the format into a format the device can readily process. This is an automated process, which the user is not aware of. Using NASUF this functionality can be performed, reducing costs by automatically extending device functionality beyond what they were initially designed to do. This provides the following features:

- Integrate the large number of services provided by devices to resolve device conflicts as and when they happen.

- Reduce the costs associated with constantly upgrading hardware solutions, when all that is required is a slight extension to the functions the device already provides.
- Devices do not have to have all the required functions, but rather can integrate and utilise third party functions provided by other devices. In this instance custom devices may be installed that provide some given function, *i.e.* information transcoding, protocol interoperability, data aggregation, or intelligent processing and reasoning.
- Devices can choose to be as thin or fat as they want and at the same time perform complex functions by loosely coupling remote services provided by other devices. This means that devices, irrespective of their capabilities (sensors, PDAs or PCs), can participate within any environment and provide and/or use the functions available.

5.4 Summary

This chapter demonstrates how our framework can be used to implement an Intelligent Home Environment, capable of interconnecting networked appliances. The case study explains how zero-configuration can be achieved and how device and service compositions can self-adapt in the advent of device or service failure. The core functions highlighted within the case study can be adapted and applied to different home networking scenarios allowing virtual appliances to be created and enabling service utilisation. Numerous configurations can be automatically created dependent on the devices and services available and the richness of the semantic service capability descriptions provided by devices. Extending the application domain further this chapter also highlights several other application domains in which NASUF can be applied.

Many lessons have been learnt through our case study with the most important being that our framework is highly flexible and portable across many different problem domains. It highlights a completely new and novel way of interconnecting and using devices that to date surpasses existing middleware solutions. By breaking the individual functions provided by devices and dispersing them within the network results in distributed networked behaviours that can be discovered and used in parallel with any other functions the device provides. It can lead to a reduction in the amount of equipment required as is the case in our medical example described above. It can also prolong the life of appliances by allowing them to extend the functions they provide beyond what they are initially designed to do. This will provide significant cost savings to consumers and forge a closer relationship between people and technology.

Technological change is about innovation. Our framework breaks operational functions down into constituent networked behaviours creating a promising foundation that aids innovation and allows new and novel solutions to be created. For example networked behaviours can be

selected and combined irrespective of what devices provide them, and new solutions can be created that could not be provided by any individual device alone, *i.e.* all the speaker functions within the network could be combined to create a virtual intercom system. The device does not explicitly exist but rather emerges for as long as the audio functions are held in an intercom configuration.

Our framework aims to solve a number of difficult challenges and although we have successfully achieved this there is still considerable room for improvement. The following Chapter provides a detailed discussion on how we implemented our framework design to realise the Case Study.

Chapter 6

6 System Implementation

6.1 Introduction

In this chapter we present the implementation for our framework described in Chapters 3 and 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 and therefore it 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 Service-Oriented Architecture

NASUF is a service-oriented architecture. It provides mechanisms that allow networked appliances to be seamlessly interconnected and offer the services they provide. Chapter 2 introduced the common concepts used within home networking, networked appliances, peer-to-peer computing and the semantic web. Throughout this chapter these concepts will be used to describe how the services that comprise NASUF realise the novel contributions detailed in Section 1.9 on page 10.

6.3 Framework Services

The following subsections discuss the implementation details for each of the services used to implement the NASUF 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 6.1 shows the services used within NASUF and the relationships that exist between them.

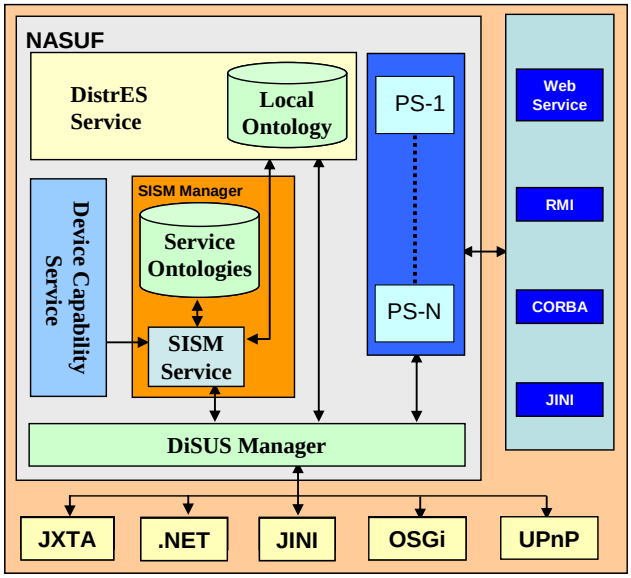

Figure 6.1 NASUF Framework

The remaining subsections discuss the key techniques used to implement NASUF which includes the JXTA peer-to-peer network; secondary and application specific services; serialisation and semantic interoperability; dynamic service composition; device capability matching; and self-adaptation.

6.3.1 The JXTA Peer-to-Peer Network

NASUF integrates heterogeneous devices; enables seamless communications; and allows services provided by devices to be shared. Within NASUF this integration is achieved using the JXTA protocols [Sun Microsystems Inc. 2005a]. These protocols allow any device to be connected to the network independent of the platform, programming language, or the transport protocols devices implement. Devices are inherently heterogeneous therefore NASUF provides abstractions that hide the underlying implementation and transport details, thus creating a logical layer whereby all devices appear homogeneous in nature. The findings of this research are that of all the current toolsets, JXTA provides the best mechanisms to achieve this (as argued in Section 2.4.8 on Page 41).

The NASUF secondary services we have developed exist within the service layer of JXTA. This allows devices to perform device capability matching; semantic service discovery; semantic interoperability; ontology evolution; dynamic service composition and self-adaptation. The NASUF secondary services extend the JXTA specifications too include these additional capabilities.

A multidisciplinary approach has been taken for inter-device communications within NASUF. The services that comprise NASUF are pre-determined and each device understands how to discover and invoke them. Pre-determined pipe advertisements are used to discover secondary

services. All devices that offer a particular secondary service use the same pipe advertisement. This ensures that devices do not continually create and publish new advertisements each time the device is connected to the NASUF network. This technique is used to minimise discovery overheads and ensure that the advertisement cache does not continually inflate over time.

Unlike secondary services, application specific services, (which are designed to publish the functions provided by devices) are numerous and the pipe advertisements used by these devices are not necessarily known by devices beforehand. As such semantic metadata is used to discover application specific services based on the behaviours they support. NASUF-enabled devices propagate messages to all devices within peer groups using the JXTA *ResolverService* protocol. This protocol allows messages to be propagated within the network, which are processed by *ResolverService* listeners implemented on devices – this provides an effective messaging system for ad hoc service discovery. Devices discover application specific services using a query containing the handler name, routing information and the message digest. We have extended the query object provided by JXTA for *ResolverService* communications to include additional XML tags that describe both the required capabilities the candidate device must support and the service behaviours the querying device requires.

The device capability tags are used to describe CPU, memory, and networking capabilities for example. This is an important requirement because the same type of service, for example an audio service, could potentially be provided by multiple devices within the NASUF network. As such the device capability model is used to select the device that can execute the service most effectively. Devices that receive query objects use the device capability tags to determine whether the capabilities it supports match or surpass the actual capabilities the device requires. Device capability models in NASUF are serialised using the CC/PP specification [Klyne 2004].

The service capability model, used in conjunction with the device capability model, semantically describes each of the functions the service provides. This allows devices to overcome the limitations associated with attribute-value pair matching to describe services in more detail. Service capability models in NASUF are serialised using the OWL-S specifications. These specifications have been used to extend the current discovery specifications provided by JXTA to enable services to be matched semantically.

6.3.2 Secondary and Application Specific Services

All the services within NASUF, whether they are secondary, such as DistrES, or application specific such as Audio or Video, are created and published as advertisements using JXTA. We have developed a service factory, which acts as a wrapper around existing JXTA services

which includes our NASUF services. Discovering these services simply requires the device to search for the service advertisement by name and extract the pipe advertisement it contains before binding to and using it. This differs from application specific services because such services are plugged into the framework by device manufacturers in order to allow access to the functions provided by devices. Consequently, equipping a device with every variation of the services contained within the network is not practical. As such application specific services are discovered using semantic discovery mechanisms provided by NASUF.

We have extended the JXTA service advertisements to include the Peer ID. This could have been overcome using the JXTA Peer Advertisement specification, however to reduce the number of discovery requests made a decision was made to place the Peer ID in the service advertisements. This allowed us to make one single discovery request for all the required information needed. If we did not do this we would have had to develop the software to find Peer advertisements as well as the service advertisements. This would require making two advertisement requests, resulting in increased network traffic and computation. Our rationale was that devices of varied capabilities will use the NASUF framework, consequently minimising the amount of traffic and the computation required would ensure that devices with limited capabilities are not over taxed.

Using the Peer ID is an important design decision, which ensures that, although more than one service may exist of the same type, devices only bind and use the service initially discovered when a connection request to the service is made. This makes sure that other pipe listeners for a pipe advertisement do not receive and process messages not destined for them. The decision to adopt this technique was based on a number of undesirable results we encountered within our implementation, whereby connection requests could be made to any pipe at the same time irrespective of the initial device and service discovered.

6.3.3 Serialisation and Machine-Processable Semantics

NASUF provides mechanisms that enable zero-configuration between devices based on capability matching. Ontological structures are used to describe what devices want and what they provide. Again a number of approaches have been considered for ontological processing and several working prototypes have been developed within this research using OpenCyc, XOL, RDF, RDF-S DAML+OIL, OWL, Jena and the Protégé-OWL API. Although, ontologically, OpenCyc provides considerable inferential capabilities it is very resource heavy to implement (120 megabyte API). Furthermore the underlying knowledge base uses a propriety language called CycL, which is somewhat restrictive because it is not considered an open standard. XOL is considered a legacy ontology language, thus is has little support in terms of tools and usage. RDF and RDF-S are W3C recommendations, consequently there is a

great deal of support and a large number of tools exist for RDF-based processing. However the expressiveness of RDF-based serialisations is limited and in most cases inferior to other ontological languages such as OWL. Within NASUF the goal is to enable devices to reason over expressive ontology serialisations and deduce not only explicit, but implicit concepts derived from atomic and complex concept compositions. In NASUF the OWL-DL sublanguage of OWL has been adopted to achieve this because a large number of reasoners exist capable of processing DL-based ontologies. This version of OWL also provides a constrained, but expressive, language that can describe rich ontological structures and at the same time support formal reasoning, consequently every device within NASUF creates and evolves OWL-DL serialisations.

OWL-DL serialisations are processed using the Protégé-OWL API [Stanford University 2005a], which overcomes the proprietary nature of OpenCyc by supporting open standards. The Protégé-OWL API is an open source project, designed to provide tools capable of processing language-neutral ontologies. This API fully supports the OWL-DL specification and is a well developed tool that has a large number of academic and industrial supporters. The API is comprehensive and progressing at a fast pace. In our implementation the Protégé-OWL Reasoner API [Stanford University 2005b] is also used, which supports several DIG compliant reasoners such as Racer [Haarslev 2001], FaCT [Horrocks 2005] and FaCT++ [Tsarkov 2005]. We have used the Racer reasoner because of its adoption within the wider research community, thus more support, tools and usage scenarios are available.

We have also carried out extensive research using the Jena API, which provides several internal and external reasoner interfaces, however a number of performance problems where encountered. For example when an inferred model is created using internal and external reasoners, out of memory errors occur. Through experimentation this limitation was overcome using the Protégé-OWL API and Racer. Jena is however used to perform simple querying on OWL-S serialisations because they are not DL compliant. This is achieved using the ontology models provided by Jena and RDQL.

Our DistrES service has been developed in Java and is used to determine if semantic relationships exist between different vocabularies. It performs hierarchical analysis via subsumption as well as equivalence and restriction checking between different concepts. The DistrES service is capable of determining whether any two concepts are disjoint from each other and can perform classification based on the properties a particular class supports. This means that the reasoner can determine what concept(s) a particular individual or class belongs to by analysing the properties it supports. This is an important requirement because services are dynamically composed by matching signatures contained in the service interface, *i.e.* the inputs and outputs used to represent a signature. This service provides a flexible abstraction

layer that enables open standard serialisations, such as OWL, to be processed and reasoned over within our NASUF implementation. DistrES uses custom algorithms we have developed in Java that utilise the functions provided by the Protégé-OWL and Racer APIs. The DistrES service extends the discovery mechanisms provided by JXTA, to enable semantic service discovery. This allows devices to more accurately discover and use services based on semantic mappings between high-level semantic descriptions of what the service does and low-level service interfaces used to bind to and invoke the service.

6.3.3.1 Describing Services Semantically

NASUF uses semantic information for service descriptions and service requests. These descriptions are serialised using OWL-S. OWL is used to serialise domain knowledge and help perform interoperability between different terminologies used in service requests and service descriptions. The OWL-S specification is in the early stages and to date is not a recommended standard. It still has a number of issues, most importantly it does not conform to OWL-DL, which makes it difficult to use with the Racer reasoner. However, the specification provides an effective and promising mechanism for describing services semantically and building a foundation on which to build.

Each application specific service within NASUF is described using OWL-S. The Service Profile is used to describe both the service request and the high-level semantics of the service. Semantically matching service requests with service advertisements is performed using the SISM service which we have developed in Java and plugged into the JXTA service layer. This service uses the *AbstractMatcher* algorithm we have developed to match the IOPEs in the service request with IOPEs described in the service advertisement. Ambiguities between different terms are resolved using the DistrES service. In conjunction with the *AbstractMatcher* algorithm the *ConcreteMatcher* algorithm we developed maps the high-level semantic descriptions defined in the Service Profile to concrete bindings within the service interface. NASUF uses WSDL to syntactically describe low-level service signatures, irrespective of the service technology being used. Through experimentation WSDL provides a specification, which is a well understood standard recommended by the W3C. This specification is flexible and extensible, allowing any service interface to be described at the syntactic level. However WSDL does not address the semantics of information. Consequently it is difficult to assess the capabilities services provide by looking at the interface alone. As such WSDL is used in conjunction with OWL-S and embedded within JXTA service advertisements to enable syntactic and semantic analysis. This extension allows devices to process service advertisements and reason about service capabilities to determine if the service provides the required behaviour.

6.3.3.2 Evolving ontological structures using general consensus

We have developed custom algorithms in Java to evolve ontological structures over time, which we have implemented in the DistrES service. The Evolutionary Pattern Extraction (EPE) algorithm allows concepts of various depths to be extracted from a device's domain ontology. The EPE extracts conceptual information from separate ontological structures using statistical analysis. Ontological structures themselves are discovered within the network using JXTA and custom queries that define the concept required. The EPE extracts commonalities from *n* ontological structures, where *n* is the number of ontology structures returned from the network, to produce an optimal structure based on general consensus. Optimal structures are merged with the device's local ontology using the Merge Algorithm (MA) that we have developed. An assumption has been made that small device specific ontologies with be developed by device manufacturers, however once the device is deployed, ontologies will be evolved and managed by NASUF using the EPE and the MA.

6.3.4 Dynamically composing services using ontology

The SISM service we developed has been implemented within NASUF allowing devices to determine what services they can form relationships with. Device manufacturers can retrieve predefined semantic descriptions and use them to find any dependency services the device requires. Service requests are described in terms of the *inputs* the service requires, the *outputs* it generates, the *preconditions* that must be satisfied and the *effects* that happen as a result of executing the service. All service requests are propagated within the network using DiSUS. Devices capable of processing requests extract the semantic information and match it against the semantic descriptions used to describe each application specific service the device provides. SISM uses the *AbstractMatcher* and *ConcreteMatcher* algorithms to achieve this.

6.3.5 Formally describing device capabilities using MAUT

A number of experiments have been performed using the MAUT formula and the CC/PP standard to calculate capability scores. Initial prototypes demonstrate that using MAUT allows NASUF to effectively evaluate device capabilities. The CC/PP specification is used as a base device capability model, which we have extended to include the MAUT constructs. The DeCap service implements the MAUT algorithm we have developed, which is used to provide an overall evaluation of the device's capabilities in conjunction with the device capability model embedded in the service request. If the device capability model score is equal to or greater than the score calculated for the device capability model extracted from the service request, then the device is said to be capable of executing the service in conformance with the querying device's requirements.

Based on several prototypes we have developed, the CC/PP specification and the MAUT algorithm provide an effective mechanism for selecting devices and services. The DeCap service is plugged into the service layer of JXTA and is used to extend the current JXTA specification to consider how capable devices are before selecting a service it provides. For example, several devices may provide "visual" services, however some devices may be more capable of processing video content than others. The current version of JXTA does not provide any mechanisms to achieve this.

6.3.6 Self-adaptive middleware

NASUF provides mechanisms that allow devices to form relationships with other devices and services within the network. When a device is initially switched on it automatically discovers the dependency services it requires. This may result in several services that provide the same functionality. Devices store each response received from within the network and use a control mechanism to adapt a particular service composition during execution. In NASUF, mechanisms are provided that allow device manufacturers to decide how service advertisements are stored and managed. In our implementation advertisements are processed in memory, consequently when the device is switched off the advertisements are lost and must be re-discovered again. However in real-world implementations some backend store, for example a database system, may be used. This may not always be the case as the environments in which these appliances exist are highly transient.

NASUF always picks the services that provide the best quality of service. If a service fails the next best service is selected and plugged into the composition. In the advent of the failed service becoming available again, it is used to replace the existing service in the composition if it improves the overall quality of the composition. This is achieved using a custom control mechanism we developed, which is implemented in the *Device* abstract class. This functionality was required because JXTA does not provide any control mechanisms to allow devices to automatically reconfigure in the event that services become unavailable. Our self-adaptation mechanism has addressed this limitation to allow compositions between devices and services to be automatically reconfigured without any human intervention as and when service failures occur.

In the remaining sections the implementation details for each of the services that comprise the NASUF architecture are discussed in more detail.

6.4 The Framework Prototype

In order to evaluate our framework design presented in Chapter 3 and 4, a prototype has been developed. This is in accordance with the case study presented in Chapter 5, which is an

Intelligent Home Environment. The prototype uses four wirelessly connected computers to simulate two televisions, a Media player and two audio speaker systems. The televisions host 'Visual' services, which process visual data streams. The Media player hosts a 'Player' service which outputs MPEG1 multimedia data, and finally the audio speaker systems host 'Audio' services, which process audio data streams.

Communication between devices is achieved using the wireless 802.11g standard and OWL-S service requests are propagated between devices in the network using the JXTA *ResolverService*. Each device implements DiSUS and either implements the SISM, DistrES and DeCap services or discovers and uses these services remotely within the network. When devices are initially switched on and have published the services they provide they automatically try and discover devices within the environment they have a relationship with. For example when the Media player is switched on it tries to discover devices capable of processing audio and video streams outputted by the player. Using a simple control interface, as illustrated in Figure 6.2, users can discover, use and control any device connected to the network and the services it provides. Note in this instance devices themselves may control other devices they have relationships with without any human intervention. For example if the user sends a "Play" command to the Media player, the player interacts and controls the speaker system and television automatically.

Using the user interface users can select the device and service capability models describing the quality of service factors the device must support and the service functionality required. These models are serialised as XML and are appended to a service request before being propagated within the network using the "Send Query" button.

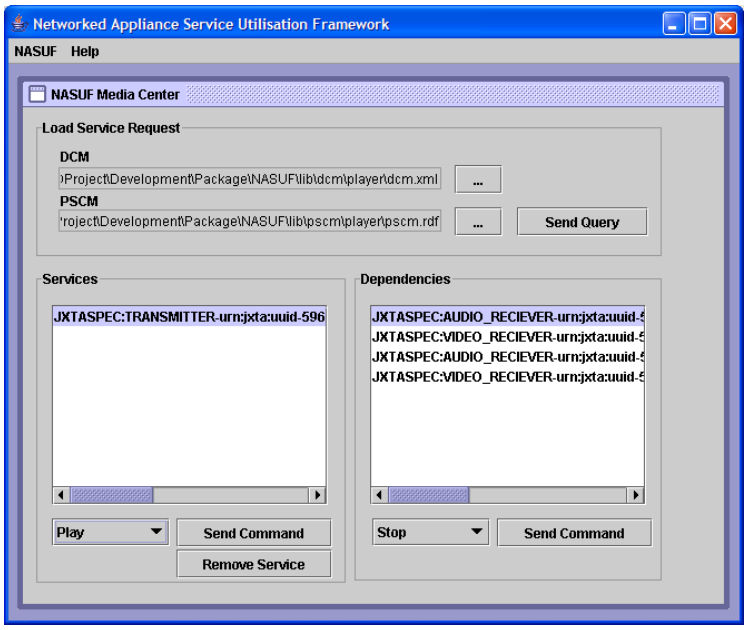

Figure 6.2 NASUF User Interface

Three tests have been developed to evaluate NASUF. The first test demonstrates that NASUF can allow devices to form relationships with other devices in the network without any human intervention. The second demonstrates that conflicts within signature mappings can be resolved using intermediary services and the last demonstrates that devices can self-adapt in the event of any device or service becoming unavailable. In the first test the Media player is started and two service requests are created using the OWL-S Service Profile. These service requests are used to find devices capable of processing audio and video streams. The Media player propagates the requests within the network using the DiSUS Manager and adds any responses to a table of candidate services, categorised according to the type of device or service discovered.

In the second test the user sends an *IncreaseVolume* or *DecreaseVolume* command to the speaker system (this is a dependency service used by the Media player as illustrated in Figure 6.2). To demonstrate parameter conflicts volume values are sent to the speaker system as strings, however the parameter should be of type integer. We set up a simple service on the network that performs data type conversions. Initially the speaker system receives the service request and determines that the IOPE in the service request (*IncreaseVolume*) can be matched with the IOPE in the service description (*IncreaseVolume*) however when the data types associated with the IOPEs are processed, the SISM service determines that the data type associated with the *IncreaseVolume* parameter in the service request is of type *String* and that the parameter *IncreaseVolume* in the service description is of type *Integer*. In this instance SISM tries to find an intermediary service capable of performing the conversion. SISM reformulates a service request, which defines two IOPEs – the first IOPE is the conflicting Input (*string*) found in the service request and the second IOPE is the required output needed to resolve the conflict (*integer*). SISM then propagates the service request using DiSUS, which is received and processed by the data type conversion service. This service takes as input a *StringValue* of type *String* and outputs an *IntegerValue* or type *Integer*. The service matches the IOPEs at an abstract and concrete level and successfully creates the extended interface metadata file and returns it to the audio speaker system. The audio speaker system stores the metadata file along with a unique ID and creates its own extended interface metadata file that links to the extended interface metadata file for our data type conversion service using the unique ID, which is then returned to the Media player.

We where able to invoke the *IncreaseVolume* command and demonstrate how the speaker system uses our intermediary data type conversion service to convert the *String* value into an *Integer* value, by substituting the conflicting parameter with the result before invoking the *IncreaseVolume* command on the audio speaker system. This is a simplistic demonstration

that only considers one parameter and simple data types however the mechanisms illustrate how conflicts can be resolved.

The third test case demonstrates how devices adapt to device and service failure. When the user sends a *Play* command, the player instructs the audio speaker system and the television to begin processing the media streams sent from the player. For demonstrative purposes the current audio speaker system being used was removed from the network to test NASUFs self-adaptation capabilities. In this instance the Media player senses this change and automatically uses a previously discovered audio service. The player binds to the audio service and instructs it to begin processing the audio data outputted by the player. To further demonstrate the adaptation mechanisms in NASUF, the previous audio speaker system used was re-published within the network. The Media player successfully senses this change and compares the device capability model for this speaker system with the device capability model for the current speaker system being used. It discovers that the newly published speaker system provides a better auditory experience than the speakers currently being used and as such it instructs the audio speaker system being used to stop processing the audio stream and instructs the newly published audio speaker system to begin processing the audio stream.

6.4.1 Technical Description

Each device publishes its functions as JXTA Peer services and allows devices within the P2P network to discover and use them. The services have been developed as JXTA Peer services, however any service technology could be used such as GLUE-STD [WebMethods 2003], which are W3C compliant Web Services.

A typical device and service capability model used to discover a device capable of processing an audio stream is illustrated in Figure 6.3 (a) and (b). The device capability model describes the capability parameters, which also includes the MAUT values. The peer service capability model describes two *inputs* which are *stop* and *listen* used to turn the speaker system on or off. It has one *output* which is a *RadioWave* indicating the type of data this device outputs. It has one *effect* which states that when the device is in use it is receiving a digitised wave and one *precondition* which states that the device should be an *AudioSpeaker*. The device and peer service capability models, in part, form the basis for service requests in NASUF.

```
<?xml version="1.0"?>
<rdf:RDF>
  <rdf:Description rdf:about="http://www.livjm.ac.uk/dcm#power">
    <dcm:importanceRating>40</dcm:importanceRating>
    <dcm:statusAssessment>Average</dcm:statusAssessment>
    <dcm:statusRating>50</dcm:statusRating>
    <dcm:importanceRanking>4</dcm:importanceRanking>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.livjm.ac.uk/dcm#MyProfile">
    <ccpp:component>http://www.livjm.ac.uk/dcm#Memory</ccpp:component>
    <ccpp:component>http://www.livjm.ac.uk/dcm#Bandwidth</ccpp:component>
    <ccpp:component>http://www.livjm.ac.uk/dcm#CPU</ccpp:component>
    <ccpp:component>http://www.livjm.ac.uk/dcm#Power</ccpp:component>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.livjm.ac.uk/dcm#Power">
    <ccpp:defaults>power</ccpp:defaults>
    <rdf:type>HardwarePlatform</rdf:type>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.livjm.ac.uk/dcm#cpu_load">
    <dcm:importanceRanking>4</dcm:importanceRanking>
    <dcm:statusRating>50</dcm:statusRating>
    <dcm:statusAssessment>Average</dcm:statusAssessment>
    <dcm:importanceRating>40</dcm:importanceRating>
  </rdf:Description>
  ….
</rdf:RDF>
```

a.

```
<profileHierarchy:ServiceRequest rdf:ID=
                    “AudioServiceRequest">
      <profile:hasInput rdf:resource=
                    "http://www.livjm.ac.uk/ServiceRequest.owl#RadioWave"/>
   <profile:hasInput rdf:resource=
                    "http://www.livjm.ac.uk/ServiceRequest.owl#Stop"/>
   <profile:hasInput rdf:resource=
                    "http://www.livjm.ac.uk/ServiceRequest.owl#Play"/>
   <profile:hasOutput rdf:resource=
                    "http://www.livjm.ac.uk/ServiceRequest.owl#RadioWave"/>
   <profile:hasEffect rdf:resource=
                    "http://www.livjm.ac.uk/ServiceRequest.owl#ReceivingAWave"/>
   <profile:hasEffect rdf:resource=
                    "http://www.livjm.ac.uk/ServiceRequest.owl#PropagatingAWave"/>
</profileHierarchy:ServiceRequest>
```

b.

Figure 6.3 NASUF Service Request Models

When a service is matched and the device providing the service has the required capabilities to effectively execute it, the service advertisement is added to the devices collection of matched services. Once all the required services have been found the device remains in an idle state until it is controlled by the user via the user interface illustrated in Figure 6.2. In this instance the user selects the required command from the drop down box located next to the *Send Command* button, which is extracted from the service interface (in this case a WSDL file – WSDL files are processed using GLUE-STD [WebMethods 2003]).

Service requests are propagated between devices in the P2P network using the JXTA *Resolver* service and processed using two event handlers called *processQuery* and *processResponse*.

All devices have a JXTA interface that allows them to join the default *peergroup* called *NetPeerGroup*. The code to achieve this is illustrated in part in Figure 6.4.

```
public void startJxta(){
    try{
        peerGroup = PeerGroupFactory.newNetPeerGroup();
        AbstractService.setPeerGroup(peerGroup);

        resolverSvr = peerGroup.getResolverService();
        resolverSvr.registerHandler(handlerName,
                    (QueryHandler)ResolverMsgHandlerFactory
                                    .createDISUS_Handler(this));
    }catch(PeerGroupException e){
        if(NASUFLogger.isEnabledFor(Level.ERROR))
            NASUFLogger.error("DiSUS: startJxta: " + e.toString());
        System.exit(1);
    }
}
```

Figure 6.4 Joining the P2P Network using JXTA

Once a device joins the peer group and registers a message handler with the *Resolver* service it can send and receive messages. Each device in the prototype registers to receive *DiSUS* messages, which are encapsulated using JXTA-defined messaging objects called *ResolverQueryMsg* and *ResolverResponseMsg*. Along with other information, OWL-S service requests we developed are wrapped in JXTA message objects and propagated within the P2P network.

Devices communicate with secondary services such as SISM and DistrES using bidirectional pipes called *BiDiPipes* in JXTA. Figure 6.5 illustrates in part how DiSUS binds to *BiDiPipes* in NASUF. All the queries used to process the service ontologies are performed using the RDQL API provided by the Jena 2.3 API.

Using a sample service request as illustrated in Figure 6.3 above, the RDQL query defined in Figure 6.6 (a) can be executed using the sample code illustrated in Figure 6.6 (b), using Jena to extract the defined inputs. The common keywords found in SQL such as *Select*, *Where*, *For* and *Using* as illustrated in Figure 6.6 (a) are also used in RDQL. Jena provides a comprehensive API that makes querying any RDF-based model, an easy process.

```
public void run() {
    pipe = disus
        .bindToService(disus.discoverCoreService(
            DistrESConstants.DISTRES_SPEC).toString(), this);

    if(!pipe.isBound()){
        if(NASUFLogger.isEnabledFor(Level.INFO)){
            NASUFLogger.info("Failed to Connect to Pipe");
        }
        return;
    }

    Message dcmMsg = new Message();
    dcmMsg.addMessageElement(ServiceDescriptionConstants.NASUF_NAMESPACE,
        new StringMessageElement("DistrESRequestType", "SemInterop", null));

    dcmMsg.addMessageElement(ServiceDescriptionConstants.NASUF_NAMESPACE,
        new StringMessageElement(DistrESConstants.X_TERM, srTerm, null));

    dcmMsg.addMessageElement(ServiceDescriptionConstants.NASUF_NAMESPACE,
        new StringMessageElement(DistrESConstants.Y_TERM, spTerm, null));

    if(NASUFLogger.isEnabledFor(Level.INFO))
        NASUFLogger.info("Sending DistrES Message");
    try{
        pipe.sendMessage(dcmMsg);
        Thread.sleep(5000);
        pipe.close();
    }catch(Exception e){
        if(NASUFLogger.isEnabledFor(Level.ERROR))
            NASUFLogger.error("AMatcher_DECAP_Handler: run: " + e.toString());
        try{
            pipe.close();
        }catch(Exception ioe){
            if(NASUFLogger.isEnabledFor(Level.ERROR))
                NASUFLogger.error("AMatcher_DECAP_Handler: run: " + ioe.toString());
        }
    }
}
```

Figure 6.5 Binding to Secondary Services

```
SELECT ?input WHERE (?x profileHierarchy:ServiceRequest ?y),
    (?y profile:hasInput ?z),
USING profile FOR
    "<http://www.daml.org/services/owl-s/1.0/Profile.owl>"
```

a.

```
public QueryResults executeQuery(OntModel ontModel, String queryString){
    Query query = new Query(queryString);
    query.setSource(ontModel);
    QueryExecution qe = new QueryEngine(query);
    QueryResults result = qe.exec();
    return result;
}
```

b.

Figure 6.6 RDQL query execution

In the prototype RDQL is used extensively to extract IOPEs and information that link the service ontologies together. The SISM algorithm uses RDQL queries in conjunction with the DistrES ontology to determine the relationships that exist between different terms. The service request IOPEs and the service description IOPEs are extracted using RDQL queries and relationships between the terms are determined using the DistrES ontology providing an effective mechanism for semantic interoperability.

When a service request is received from a device, DiSUS attempts to match the service request against the Service Profiles for each application specific service it provides. This is achieved using the SISM service. Resolving ambiguities between terms that are syntactically distinct but semantically equivalent is achieved using the DistrES service which uses an OWL ontology [W3C 2004] we developed for networked appliances as illustrated, in part, in Figure 6.7 – more example models of the ontology can be seen in Appendix D on Page 230.

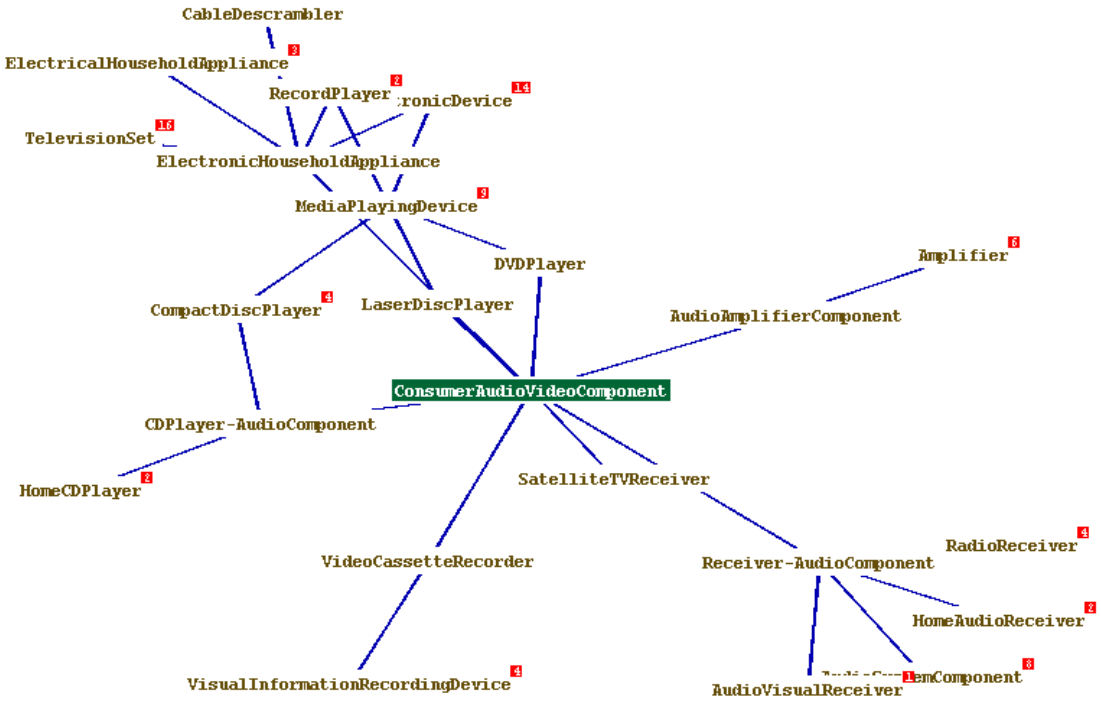

Figure 6.7 DistrES Networked Appliances Ontology

The ontology itself conforms to the OWL-DL language [W3C 2004] and currently has about 500 concepts that semantically describe common household appliances and their associated properties such as inputs outputs and events. The ontology was developed using the Protégé 3.1 ontology editor and the OWL plug-in [Horridge 2004]. The domain ontology allows devices to determine if any terms are conceptually related. In the implementation the Protégé-OWL API is used to load and process the ontology.

Domain knowledge is evolved using the DistrES service based on general consensus. Figure 6.8 provides, in part, the code used to extract the top *n* nodes, where *n* is the number common nodes that exist within all ontology structures received from the P2P network. This is a configurable feature that is dependent on the application. Class and relationship selection can be based on manual configuration or using some automatic feedback mechanism implemented as a service in NASUF.

The Protégé-OWL API provides all the common methods required to reason over OWL-DL serialisations. It also provides methods that allow the properties of concepts to be reasoned

over and it allows inferred knowledge structures to be calculated. Figure 6.9 illustrates some of the code used in SISM to determine if a subclass or subsumption relationship exists between two concepts.

```
private Object getTopClasses(int topClasses){
    Object tempKey = null;
    Object tempValue = null;
    Map topClassesCollection = new TreeMap();
    if(topClasses < classF.size()){
        for(int i = 0; i < topClasses; i++){
            int count = 0;
            Iterator iter = classF.keySet().iterator();
            while(iter.hasNext()){
                Object cls = iter.next();
                int value = ((Integer)classF.get(cls)).intValue();
                if(value > count){
                    tempKey = cls;
                    tempValue = classF.get(cls);
                }
            }
            if(tempKey != null && tempValue != null){
                classF.remove(tempKey);
                topClassesCollection.put(tempKey, tempValue);

            }
        }
        return topClassesCollection;
    }else{
        return classF;
    }
}
```

Figure 6.8 Extracting the Top n Classes

```
//This method returns a true or false value depending on whether
//class1 is a subclass of class2.
public boolean isSubclassOf(Object class1, Object class2){
    Collection col = this.getSubclasses(class2);
    if(col.contains(
        distresOntology
            .getOWLNamedClass(
                (String)class1))){
         return true;
    }else{
        return false;
    }
}
```

a.

```
public boolean isSubsumedBy(Object class1, Object class2) {
    try{
        return reasoner
            .isSubsumedBy(
                distresOntology
                    .getOWLNamedClass((String)class1),
                distresOntology
                    .getOWLNamedClass((String)class2), null);
    }catch(Exception e){
        if(NASUFLogger.isEnabledFor(Level.ERROR))
            NASUFLogger.error("getDescendentClasses Error: " +
                e.toString());
            return false;
    }
}
```

b.

Figure 6.9 Reasoning over the domain ontology

Devices self-adapt using the DiSUS manager, the registered dependency services the device has and the DeCap Service. The code in Figure 6.10 illustrates, in part, how the best service in a composition is selected when conflicts are encountered.

```
protected String selectBestService(List serviceCollection){
    IDataObject bestService = null;
    IDataObject tempService;
    double dcm_score = 0.0;
    Iterator iter = serviceCollection.iterator();
    try{
        while(iter.hasNext()){
            tempService = (IDataObject)iter.next();
            if((Double.valueOf(tempService.getDecapValue()).doubleValue()) > dcm_score){
                bestService = tempService;
                dcm_score = Double.valueOf(tempService.getDecapValue()).doubleValue();
            }
        }
    }catch(Exception e){
        if(NASUFLogger.isEnabledFor(Level.ERROR))
            NASUFLogger.error("Device: selectBestService: " +
                e.toString());
    }
    return bestService.getModuleSpec();
}
```

Figure 6.10 Selecting the Best Service

The application specific services used in the prototype have been developed using Java and allow audio and video to be transmitted and received between devices. These media processing services have been implemented using the Java Media Framework (JMF) Performance pack for Windows, based on version 2.1.1 [Sun Microsystems Inc. 2005b].

The NASUF implementation comprises around 120 Java classes. This totals around 15 thousand lines of Java code (15 KLOC). The implementation uses several open source Java APIs, consequently these must also be bundled with the NASUF APIs at deployment. The implementation is portable and runs on different platforms. NASUF is a service-oriented framework so depending on what secondary services devices implement also affects the size of the deployment package. For example if a device does not implement DistrES then the reasoner and ontology processing APIs do not need to be deployed on the device. This ensures that devices irrespective of there capabilities can use and operate within the NASUF network. The NASUF application was deployed using ANT [Hightower 2002], which is a tool used to create and set-up deployment configurations.

6.4.2 Prototype Configuration

In order to evaluate the NASUF implementation, a prototype was set-up within the School of Computing and Mathematical Sciences at Liverpool John Moores University. This prototype was set-up as a distributed service-oriented architecture on top of a wireless network. The configuration consisted of the following off-the-shelf components:

- A Cabletron Smart Switch Router 2000
- Entrasys Roamabout Access Point

- RoamAbout 802.11g PCMCIA network cards
- Four wirelessly connected Intel Pentium 4 – 1.8 GHz machines running Windows XP Professional, Service Pack Two, with 500 megabytes of RAM.

Several environment parameters where considered to run a real-world test and demonstrate the key functions the NASFUF framework provides. These scenario parameters are detailed in Table 6.1.

Scenario Parameters	
Network	
Transmitter Range	100 Meters
Bandwidth	54 Mbps
Number of Nodes	4
Pack Size	2048 bytes
Environment Size	100x100 Meters
Software	
OS	Windows XP Service Pack 2
Java	1.4.2_06-b03
JMF	2.1.1e
JXTA	2.3.1
OWL-S	1.0
Jena	2.0
Protégé-OWL API	2.1
Prototype	
Running Time	4 Minutes
Protocols	802.11g
Media Transmitted	MPEG1 Video (JPEG/RTP)

Table 6.1 Scenario Parameters

All the machines used within the prototype test-bed where connected using the standard TCP/IP protocol. The 1.4.2_06-b03 version of the Java Development Kit was used on all machines within the network. Several decisions where made regarding this network configuration. The first decision being that all devices must be connected using wireless communications. The second decision was that the 802.11g 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.

6.4.3 System Operation

To test the operational capabilities, all devices implemented and published all the secondary services that comprise the NASUF framework. Each device also publishes the application specific services it provides. For example, the television device publishes audio and video services. Devices that require dependency services begin by trying to discover services based on the behavioural functions they require. For example, the Media player begins by trying to find audio and video services provided by devices capable of processing the multimedia

streams the Media player outputs. Once devices have published and run all services they remain in an idle state until they are controlled via the NASUF user interface.

Using the user interface we tested whether our prototype could discover devices and services using a number of device and service capability models. For example, we tested the discovery of television services by manipulating the details described in the device capability model, *i.e.* specified that devices must have low, medium and high capabilities. We also tested that our prototype could pin-point application specific services using the semantic descriptions contained in the service capability model. Our implementation illustrated that this could be effectively achieved.

When all devices where in an idle state, using the user interface we discovered a Media player and instructed it to play a movie. Using the quality of service features supported within NASUF, our framework was capable of selecting the best visual and audio services within our network configuration. We further demonstrated that devices could self-adapt when environmental changes where encountered. We achieved this by removing devices from the network during execution to see if alternative devices could automatically be discovered and plugged into the composition with minimal disruption. For example, when we removed the visual service from the composition, the Media player automatically discovered and invoked an alternative visual service. The prototype also demonstrated that when the better visual service came back on-line again it could successfully revert to this previous service to improve the composite solution. Overall the operational functionality exhibited by our prototype illustrated that secondary and application specific services could be seamlessly integrated and removed from the network without disrupting service compositions.

Furthermore, the secondary services that comprise NASUF are optional, *i.e.* devices are not required to implement them. We tested our implementation to determine whether devices could remain functional even though minimal secondary services where available. Initially, all devices implemented and ran all the required secondary services. We began to de-register secondary services from the network provided by each device. Our prototype illustrates that even when a device de-registers its secondary services it can automatically discover the required secondary service provided by another device within the network and use it. The prototype demonstrated that all our devices could operate effectively when only one device provides a set of secondary services. Consequently this makes our implementation highly fault-tolerant whereby devices only fail to function when no secondary services are available.

6.5 Summary

This chapter has described the main implementation details used to evaluate our NASUF framework. It discussed and argued the tools and standards that we have used and highlighted

where existing tools have been extended to realise our novel contributions. Although devices are required to implement the DiSUS manager they are free to explicitly implement the remaining secondary services or discover and use these services provided by other devices within the network. This provides considerable fault-tolerance through secondary service replication. This chapter also illustrated how annotating service descriptions and service requests using semantic serialisations provides a more effective mechanism for matching services more accurately.

Many aspects of the design have been implemented, which includes the service-oriented architecture and mechanisms to publish secondary and application specific peer services. Services can be discovered based on semantic descriptions and ambiguities between domain knowledge can be resolved using distributed device ontologies based on general consensus. Services can be discovered based on capability matching rather than attribute-value pair matching, which allows for greater flexibility and a more inclusive range of query possibilities.

Devices can form dynamic compositions between services contained within the network using semantic service descriptions and can self-adapt as and when services either become unavailable or re-register themselves within the network. This chapter has also argued that devices support different capabilities and as such some devices will be better equipped to provide a given service than others. Our implementation illustrates how services are selected based on how effectively the device can execute the service.

The goal of our implementation was to demonstrate an idea and ensure that the requirements and challenges described in Chapter 1 could be addressed. It was not about delivering a final product and as such the overall performance of the implementation was not a consideration. What we have learnt from the implementation is that we are trying to solve very difficult problems, for example dynamic service composition and ontology evolution. However our goal was to address these problems head on and attempt to create a foundation on which to build. We believe that we have successfully achieved this. We have a fully working prototype that demonstrates the key novel contributions made within this thesis.

We have learnt that there are several grey areas within our research that are dependent on numerous factors. As with P2P implementations, whether or not particular content can be found is dependent on the number of nodes connected within the network and how many people hold the content sought after. This is the same with our approach whereby success is dependent on the number of devices connected to the network and the total number of services and semantic data used to describe and discover services. This said, P2P is becoming a networking model of choice and it is envisaged that networked appliances will be firmly

embedded within such a networking model. Sound business models and user acceptance will be the deciding factors. The following Chapter provides a qualitative evaluation of our design.

Chapter 7

7 Evaluation

7.1 Introduction

Chapter 1 described the requirements needed to address some of the limitations with current networked appliances and home networking approaches. These requirements detail what is needed to enable flexible appliances and middleware solutions that will allow networked devices to automatically configure and re-configure and self-adapt over time. Each of these requirements forms the basis for the qualitative evaluation of our proposed framework.

7.2 Service-Oriented Architecture

The key requirement was to provide an open middleware architecture that utilises open standards, promotes interoperability and disperses the operational functions devices provide within the network as independent services. In doing so flexibility is seen as paramount, and as such, our framework ensures that functionality is readily available through secondary service replication. This idea is based on current file sharing principles whereby popular files are distributed, shared and discovered within a P2P network. Our framework adopts the same principle, however as well as content, services are also replicated. This means that even if secondary services become unavailable there may be an alternative service within the network that can be discovered and used that provides the same functionality. This makes our framework robust and highly fault-tolerant, which ensures that device and service compositions are more reliable.

This can be justified using two mathematical proofs, which illustrate serial and parallel system reliability when services are composed. In this context services are carefully choreographed in series using workflow standards [Andrews 2005] whereas parallel compositions are performed using distributed P2P techniques.

$$P(A \cap B) = P(A \cdot B) \tag{1}$$

$$R_i = P(A_i) = p_i \tag{2}$$

$$Q_i = P(\overline{A}_i) = 1 - p_i = q_i \tag{3}$$

$$\begin{aligned} R_s &\equiv P(A_1 \cdot A_2 \ldots A_n) \\ &= P(A_1)P(A_2 \mid A_1)\ldots P(A_n \mid A_1 A_2 \ldots A_{n-1}) \\ &= P(A_1)P(A_2)\ldots P(A_n), \textit{if independent} \\ &= \prod_{i=1}^{n} P(A_i) \\ &= \prod_{i=1}^{n} R_i \end{aligned} \tag{4}$$

$$Q_s = 1 - R_s = 1 - \prod_{i-1}^{n} R_i = 1 - \prod_{i=1}^{n} (1 - Q_i) \tag{5}$$

Figure 7.1 Serial Service Reliability

In Figure 7.1 equation (1) defines the set theory representation for sequential reliability of service compositions. In this instance the probability of *A* intersection *B* is equal to the probability of *A* multiplied by the probability of *B*. Equation (2) describes the reliability of individual services, where R_i is an individual reliable service within the service space and p_i is the probability value indicating how reliable the service is. Equation (3) describes the unreliability of an individual service, where Q_i is an individual unreliable service within the service space and q_i is the probability value describing how unreliable the service is. Equation (4) describes the system reliability, which is the joint probability of all services in the composition. Finally equation (5) describes the unreliability of the system.

To take an example, assume we use three services and each service has a reliability value of 90% then the following probabilities can be calculated.

Individual service reliability: $P(A_i) = R_i = p = 0.90$

Unreliability of individual service: $Q_i = 1 - R_i = 0.10$

System Reliability: $R_s = 0.90 * 0.90 * 0.90 = 0.729$

System unreliability: $Q_s = 1 - R_s = 1 - p^3 = 1 - (0.90)^3 = 1 - 0.729 = 0.271$

Now that we have values for the reliability of serial service composition we can compare this with the reliability of a system that uses parallel service composition, as is the case with service-oriented architectures based on P2P concepts.

$$P(A \cup B) = P(A + B) \tag{1}$$

$$R_i = P(A_i) = p_i \tag{2}$$

$$Q_i = P(\overline{A}_i) = 1 - p_i = q_i \tag{3}$$

$$\begin{aligned} R_s &\equiv P(A_1 + A_2 + \ldots + A_n) \\ &= 1 - (P(\overline{A}_1) * P(\overline{A_2}) * \cdots * P(\overline{A}_n)) \\ &= 1 - P(\overline{A}_1)P(\overline{A}_2)\ldots P(\overline{A}_n), \textit{if independent} \\ &= 1 - \prod_{i=1}^{n} P(\overline{A}_i) \\ &= 1 - \prod_{i=1}^{n} Q_i \end{aligned} \tag{4}$$

$$Q_p = \prod_{i-1}^{n} Q_i \tag{5}$$

Figure 7.2 Parallel Service Reliability

In Figure 7.2 equation (1) defines the set theory representation for parallel reliability of service compositions. In this instance the probability of *A* union *B* is equal to the probability of *A* plus the probability of *B*. Equation (2) describes the reliability of individual services, where R_i is an individual reliable service within the service space and p_i is the probability value indicating how reliable the service is. Equation (3) describes the unreliability of an individual service, where Q_i is an individual unreliable service within the service space and *qi* is the probability value describing how unreliable the service is. Equation (4) describes the system reliability, which is the joint probability of all components. Finally equation (5) describes the unreliability of the system.

Again, taking an example, assume service *A* has a reliability value of 90% and Service *B* has a reliability value of 80%

P(A) = 0.90 and P(B) = 0.80. If this is a parallel system them

$$\begin{aligned} P(A+B) &\equiv P(A) + P(B) - P(A \cdot B) \\ &= P(A) + P(B) - P(A) \cdot P(B) \\ &= 0.90 + 0.80 - 0.90 \cdot 0.80 \\ &= 1.7 - 0.72 \\ &= 0.98 \end{aligned}$$

System Reliability: $R_p = 0.98$

System unreliability: $Q_p = 1 - R_p = 0.2$

The redundancy of the parallel system allows either-or services to function. This results in a system that remains operational with a higher probability than individual services acting in series. In this instance, redundancy increases reliability. Successful operation of each service is independent or at least pluggable. This means that in the event of a service becoming unavailable the functionality can be automatically discovered and plugged into the composition with minimal disruption.

This level of flexibility ensures that our framework allows devices to use service functionality discovered within the network provided by either it or other devices. In order to achieve this it is important that devices are broken down into their constituent parts whereby individual functions can be replicated, accessed and used via the network. This requirement allows devices to participate with and create service-oriented applications by picking and constructing individual services to form high-level compositions.

Using parallel service composition and P2P techniques to redundantly replicate services is a new and novel approach within networked appliance and home networking research [Fergus 2003a]. Research initiatives such as OSGi, UPnP, DLNA, HAVi, VHN, PLC, ePerSpace, MediaNet and Runes to name a few primarily focus on carefully choreographing solutions using different workflow standards such as WSFL and BPEL4WS. As long as all services in the composition are available and the locations within which they reside remain the same operation remains reliable. However if any service changes in anyway, *i.e.* becomes unavailable or moves location then the whole composition may be rendered inoperable. In our framework an alternative service would be automatically discovered and plugged into the composition with minimal disruption.

Our framework differs in its ability to not only discover and use secondary services which are pre-determined, but to also discover application specific services that abstract the individual functions devices provide [Fergus 2005a]. Our framework demonstrates this using peer service capability matching algorithms, that process semantic metadata wrapped around services allowing devices to reason over what functions devices provide. High-level semantics [DAML 2003c] are mapped onto concrete signatures defined in the service interface. The signature itself is the method name along with its associated parameters and data type information. Devices use these descriptions to reason in any direction, *i.e.* from the signature to the high-level semantics or vice versa, and select functions based on the capabilities the semantic description and service interface describes. Our implementation supports this functionality and effectively performs this mapping [Fergus 2005a]. Devices propagate service requests containing the semantics that define the required behaviours a

candidate service must support. These high-level semantics are matched against semantic descriptions used to describe a service using our framework, which links semantic information in the service request with parameters contained in service signatures. Our framework services can match any service request with any service behaviour in the network as long as that behaviour exists. One possible downside relates to environments that are more ad hoc in nature. Because no control can be placed over how and what services are hosted, it could be more difficult to exactly match service request semantics with parameters in a given signature. The probability of no match occurring could be reduced by defining methods with required and optional parameters, *i.e.* create multiple methods with different parameter lengths whereby the simplest method only contains the absolute required parameters, whilst more specialised versions contain additional optional parameters.

Our framework hosts all the secondary and application specific services within the network and as such is a pure service-oriented architecture. We have extended the JXTA specification to overcome the restrictive syntactic matching algorithms used in JXTA to discover and host services. Additional services have been added to the service layer to enable devices to discover services semantically based on how capable the device is of providing the service. Another distinct feature supported by our framework and which has been demonstrated in the implementation is the ability to enable devices to automatically form compositions between devices and services without any human intervention. Again the JXTA specifications have been extended to include zero-configuration mechanisms that utilise current P2P concepts and the semantic matching capabilities provided by our framework. Services are selected based on how capable the device is. To date current service-oriented specifications do not support these functions.

Furthermore we have extended the concepts surrounding P2P, whereby we not only focus on multimedia content sharing but also on the idea of distributing and sharing services. P2P is typically associated with file-sharing, however these overlay networks can offer much more by sharing networked behaviours as services. We have clearly made novel contributions within this area and demonstrated how P2P can be used to enhance and extend networked appliances and home networking configurations [Fergus 2003a, Fergus 2003b, Mingkhwan 2004, Fergus 2005a, Mingkhwan 2005]. To our knowledge our framework is the first to use P2P techniques to disperse operational functions provided by networked appliances. We have demonstrated that this approach is feasible using our prototype, which has shown that key functions, described in this thesis and which are not provided by other approaches such as OSGi, can be realised.

7.3 Semantic Discovery

We have argued that multiple application-specific services will co-exist, albeit with different syntactic descriptions. However conceptually they may provide the same functionality. Many researchers believe that lessons must be learnt from the World Wide Web, where we are drowning in information but starved of knowledge [Naisbitt 1991]. This is directly attributed to the representation used to describe content, which is primarily human centric. Consequently developing software to read and understand Web pages is difficult. This problem has transferred itself to Web Services whereby using and composing services is primarily a human activity. McIlraith *et al.* [McIlraith 2003] state there is a need to describe Web Services in terms of their capabilities in an unambiguous, computer-interpretable language. Combining Web Service technology with the Semantic Web will allow services to be more accurately discovered, composed and executed. Only when this is achieved will we see the true potential of service technologies.

Paolucci *et al.* [Paolucci 2003] also believe the way forward for service technologies is to add semantics. They argue that we need to move away from syntactic service descriptions and discovery and instead discover services based on their capabilities. They use a term called "sufficiently similar", which, in its strongest sense states that a service description and a service request are sufficiently similar when they describe exactly the same service. They state that this is too restrictive, because advertisers and requesters have no prior agreements on how a service is presented. A restrictive criterion on matching is bound to fail to recognise similarities between service descriptions and service requests. To accommodate a softer definition of "sufficiently similar" Paolucci *et al.* explain that there is a need to allow matching engines to perform flexible matches based on the degree of similarity between the service request and the service description.

In further support of machine-processable semantics, linking all the salient headings within this section, is the work carried out by Maedche *et al.* [Maedche 2003]. They provide an assessment of service-driven systems and describe the need to converge three separate technologies – Web Services, P2P technologies and the Semantic Web. They argue that combining these technologies allow services to be identified, located and invoked. Maedche *et al.* point out that this new paradigm is important to the development of service-enabled systems, however they also state that this is no easy task and the integration process itself gives rise to new complexities such as locating and integrating services on the fly, semantic interoperability, data heterogeneity and process mediation.

Our framework presented in this thesis demonstrates that irrespective of how services are described, conceptual mappings can be determined allowing services to be selected that

support descriptions that are syntactically distinct but semantically equivalent. This is dependent on the total number of concepts shared between devices within the network. In a real world scenario, concepts will be numerous and globally distributed between millions of devices connected within the network. As such the more concepts that exist within the network the more likely semantic interoperability may be performed [Fergus 2003b].

Our framework ensures that all service descriptions and service requests are described using rich ontological constructs and ontologies are evolved over time using general consensus [Fergus 2003b]. The following formula can be used to determine the probability of selecting a concept from some sample concept space, where *n* is the number of successful outcomes and *m* is the number of possible outcomes.

$$P(E) = \frac{n}{m} \tag{1}$$

Figure 7.3 Probability of find n in set m

For example if the concept space, which may be distributed amongst numerous devices within the network, is defined as follows:

$$\Omega = \{c1, c2, c3, c4, c5, c6, c7, c8, c9, c10\}$$

The probability of finding the following concept in the global ontology

$$E = \{c5\}$$

can be defined as:

$$0 \le P(E) \le 1 \tag{2}$$

$$P(E) = \frac{1}{10} = 0.1 \tag{3}$$

Figure 7.4 Find a concept in a global ontology

If the concept *c3* and *c4* define the same concept, *i.e.* 'Audio' then the probability of finding the concept 'Audio' can be defined as

$$0 \le P(E) \le 1 \tag{4}$$

$$P(E) = \frac{1}{10} + \frac{1}{10} = 0.2 \tag{5}$$

Figure 7.5 Finding one or more concepts in a global ontology

Determining the critical mass for finding any given concept in the global concept space is dependent on the concept being searched for and the concepts contained within the concept space. If the search concept does not exist in the concept space the probability of finding the

concept is 0. If every concept in the concept space is equal to the concept being searched for then the outcome will be 1. Our framework creates a rich distributed ontology space which allows concepts to be distributed, evolved and used to aid semantic interoperability. This has been achieved using P2P concepts that utilise the replication functions. Concepts are distributed and duplicated between devices in the network. Much like current P2P implementations the more popular a particular concept is the more times it will be replicated.

Using semantic descriptions, our framework accurately discovers services by matching the capability descriptions described in both the service description and the service request. Each IOPE in the service request is matched with each IOPE contained in the service description and if all IOPEs are matched this constitutes an abstract match. Using the case study the *inputs* describe the media formats devices support, whilst *outputs* describe the type of multimedia *output*, dependent on the device. *Preconditions* are used to further constrain the type of device/service selected. For example if a multimedia player is looking for a device to process audio then the *Precondition* may be set as "AudioSink". *Effects* are used to further constrain the selected device and describe the types of effects the device/service is susceptible to. For example the effect of sending audio data to an "AudioSink" results in radio waves being outputted by the device. Conversely devices use IOPEs to describe similar services, albeit the terminology may be different, which is demonstrated in the prototype developed for the case study, where IOPEs are described syntactically different whilst retaining the same semantics. As such it becomes important to resolve any ambiguities that appear. Our framework achieves this by performing semantic interoperability between IOPEs using the device's local ontology and ontologies provided by other devices within the network [Fergus 2005a].

The semantic interoperability mechanisms within our framework provide a base solution and illustrate that high-level semantics can be mapped to low-level signatures. Our framework has the ability to evolve ontological structures without having any centralised authority. Through device-to-device communications these structures are evolved based on commonalities that exist between all concepts, relating to the structure to be evolved, within the network [Fergus 2003b]. If the device contains the concepts then differences between terms can be resolved. However, if the device needs to query the P2P network to discover the concept then this may result in delays. The factors affecting this are the number of concepts and devices that exist, and the density of the concepts themselves, *i.e*. how may classes and relationships exist within the concept. As such our framework allows device manufacturers to perform this function as a backend management task carried out when the device is idle. This feature of our framework illustrates that using P2P technologies in conjunction with general consensus mechanisms, ontological structures can be automatically evolved and managed. Consequently concepts are

not subjective because they conform to the general consensus not the subjective opinions of a single ontology engineer - the more devices that support the concept, the more prominent the concept becomes, whilst less common concepts are de-emphasised over time. This provides considerable advantages over existing ontology evolution approaches and will become increasingly more important as devices and services become more ubiquitous and ad hoc in nature.

The way our framework processes semantic data is novel. Current approaches such as PROMPT, Chimaera, and ONION rely on knowledge consortiums and to date are incapable of automating the evolution and management of ontologies. They adopt a more centralised approach whereby a single ontology is developed which all systems reference or multiple ontologies are used and connected through manual links. Our framework completely automates this process where every device is treated as a self-governing knowledge node. Our prototype demonstrates that our approach works whereby we can distribute concepts and evolve them over time without any human intervention. We have demonstrated that this works, however to date this has only been tested on simple ontology structures. To the best of our knowledge our approach is novel and is a new way of distributing and managing ontological structures devoid of centralised repositories or any human intervention [Fergus 2003b].

7.4 Device Capability Matching

As networked appliances become more widespread it will become increasingly more important to not only discover required functionality but to also select devices that can best execute that functionality. NASUF supports this requirement and ensures devices that provide the best quality of service are selected to execute a particular service. Using high-level interfaces device manufacturers can specify the key capability parameters used to assess what capabilities the device must have including their associated capability value. Our framework uses an adaptation of the Multi-Attribute Utility Theory (MAUT) [Kumar 2003] algorithm and the implementation illustrates that functionality can be selected which takes into account the devices that best execute a given service. The formula defined in Figure 7.6 calculates the percentage of a resource required, where a resource *r* offers a service *s* that requires $ac_{s,r}$ units of some total resource value tr_r.

$$resc_{s,r} = \frac{ac_{s,r}}{tr_r}$$

Figure 7.6 Percentage of resource required

This formula allows the DeCap service to determine what percentage of some resource will be used given the total value of the resource available. The DeCap service also determines if the device is overloaded by calculating how much of the available resources on average are used by the device, *i.e*. CPU usage. Furthermore it is possible that the quality of service will be affected because the computation may be shared across a large number of processes. When this is the case, DeCap calculates the overhead for each resource the service requester deems important and compares it to the desired capability defined in the service request. The DCS achieves this using MAUT. The MAUT algorithm is implemented in DeCap and is used to produce an overall capability score for some device *D* given the attributes defined in the device's DCM. This formula is defined as,

$$DCScore(D, DCM) = \sum_{i=1}^{d} cw_i(DCM) \cdot D(v_i)$$

Figure 7.7 Calculate device capability score

where *DCScore* is the overall capability score for device *D* according to the device capability model *DCM*, *d* is the number of capabilities for the type of device, $cw_i(DCM)$ is the importance rating of attribute *i* according to device *DCM*, and $D(v_i)$ is the status rating for attribute *i*. The importance rating describes how important a given attribute is in relation to all the attributes used, *e.g.* the CPU attribute may be the second most important attribute with an importance rating of 30, which means that the CPU is considered three times more important than an attribute with an importance rating of 10. The status rating describes how well the device supports a particular attribute, *e.g.* a device may have "Excellent" for its CPU attribute, which may equate to a value of 75 – therefore calculating a capability score for CPU, could be achieved by multiplying 30 * 75 which is equal to 2250.

Given the two formulas, the device calculates the service ratings programmatically by estimating the average attribute values from the operating system itself and assigning the appropriate status rating. For example, if the device uses on average 25% of its CPU when the required service is executed we may assign the CPU_Load a status assessment of "Excellent" with a status rating of 75. The equation defined in (3) illustrates that the MAUT formula has been amended to take into account the current resource load and the load required to execute a service. In this instance the *DCScore* and the $resc_{s,r}$ are added to give a combined resource load value, indicating whether the device can effectively execute a service it provides.

$$DCScore(D, DCM) = \sum_{i=1}^{d} cw_i(DCM) \cdot D(v_i) \cdot (1 - resc_{s,r})$$

Figure 7.8 Extended MAUT formula

When the terms in the DCP and the DCM are processed, any ambiguities that are encountered are resolved using the DistrES algorithm. When the formula in Figure 7.8 is used to calculate the score for the DCM, it is compared with the score generated for the DCP. If the DCM score is equal to or greater than the score for the DCP then the device is said to be capable of effectively executing the service, whilst ensuring the quality of service is maintained. In this instance the service details are returned to the service requester.

Our framework enables devices to create compositions with other devices within the network and takes into account how well the device is capable of executing the service it provides [Mingkhwan 2005]. This technique provides a mechanism that always ensures the best possible composition is available based on those devices and services that are available at any given time. This function is currently not implemented in any other middleware standards.

7.5 Dynamic Service Composition

Trying to dynamically compose services is an area of research that has received a considerable amount of interest because of the benefits it can bring [Fujii 2004]. In the Web Services community similar research is being carried out to facilitate dynamic on-the-fly service composition. This is seen as a key step towards scalable and robust Web Service frameworks. At present, current approaches to composite Web services assume a closed world; consequently all services within the composition must be predetermined. The difficulty in a real-world setting is that Web services may become unavailable and the lack of control makes it difficult to predetermine service and network capabilities. As such this may result in unpredictable results and even composite service failure.

Because of the difficulties associated with dynamic service composition, manual and semi-automated approaches still receive considerable consideration [Chakraborty 2003, Chen 2003, Sirin 2003, Sycara 2003]. This thesis opposes these approaches because they are too inflexible for innovative solutions and we argue that devices and services need to be dynamically composed on the fly based on what is available to the device at any given time. This keeps with the visions provided by Fujii et al. and Madhusundan *et al.* where devices in our framework dynamically discover, compose and execute services as and when they are required without using templates or carefully choreographed composition scripts such as the ones defined in [Leymann 2004, Andrews 2005]. In our framework devices are pre-configured with service capability requests containing the IOPE descriptions for each service the device requires. For example in our case study on Page 113 of this thesis, the Media player has two service requests – one for an audio service and one for a video service. When the device is initially switched on these service requests are propagated within the network and any matching services are found. This provides a base solution and demonstrates that our

framework can dynamically discover and loosely bind to any service within the network using pre-defined service requests. In the current implementation we have assumed that invocation methods provided by devices are operations with no parameters such as “stop”, “play” and “listen”. So although the *Inputs* and *Outputs* describe the data received and outputted by devices this in effect describes the type of information passed or received from endpoints. To enable true dynamic service composition more descriptive service ontologies need to be used and detailed signature matching needs to be performed that allows high-level semantics to be mapped to signatures in the service interface. This functionality is provided by our framework [Fergus 2005a] which maps the service ontologies to service interfaces and enables devices to dynamically compose services on the fly.

What we have found is that it is possible to automatically discover, bind to and invoke services using high-level semantics [Fergus 2005a]. The prototype demonstrates that using semantic descriptions to process services in terms of their capabilities is a viable approach and to date this is a new strand of research within networked appliances and home networking research.

Coupled with our service-oriented architecture and use of semantic metadata, our framework provides robust mechanisms that improve the overall execution of service compositions surpassing existing service-oriented architectures that use carefully choreographed composition plans.

7.6 Self-Adaptation

One of the key factors within our framework is to enable devices to form compositions and correct problems that occur automatically with minimal human intervention. Utilising advances within the area of self-adaptive software research, the vision of self-healing software forms part of our framework architecture. This is becoming an increasingly important feature of software development, Laddaga *et al.* state

“The goal of self adaptive software is the creation of technology to enable programs to understand, monitor and modify themselves. Self adaptive software understands: what it does; how it does it; how to evaluate its own performance; and thus how to respond to changing conditions.”

To further strengthen this definition the DARPA Broad Agency Announcement on Self-Adaptive Software provide the following definition

“self-adaptive software evaluates its own behaviour and changes behaviour when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible.”

Our framework implements this functionality by automatically enabling devices to form relationships with other devices when they come online [Fergus 2005a]. The effect of this is that the device is self-aware of breaks in the relationships it has with devices it has previously discovered. Any problems encountered within compositions are sensed, *i.e*. data or control pipes become unavailable – determined by periodically sending heartbeat messages to devices and services. Devices also perform cleanup procedures which inform other devices within the network when the device or any of its services become unavailable. These messages are received by devices and used to determine whether the device or the service affects the composition it is in. Furthermore these messages are processed and used to determine whether the composition of devices and services can be improved to improve the overall performance. This being the case, our framework allows devices to promote and demote services automatically as changes occur. These functions allow devices to automatically make compensatory changes as and when required and thus provide effective mechanisms for self-adaptation within networked appliance networks. This functionality is not evident in existing approaches such as OSGi, UPnP and DLNA.

7.7 Comparison with existing Approaches

In this section we compare our framework with three state-of-the-art networked appliance and home network approaches. We use our novel contributions (service-oriented networking, service discovery, device capability matching, dynamic service composition, and self-adaptation) as a basis for our comparison and compare them to the corresponding features provided by these architectures, which are Universal Plug and Play, the Open Services Gateway Initiative, and the Reconfigurable Ubiquitous Networked Embedded Systems framework.

7.7.1 Universal Plug and Play

- *Service-Oriented Networking* – UPnP is a service-oriented architecture that provides mechanisms to disperse device functions within the network in the same way our framework does. The main limitation with UPnP however is its inability to provide or access services outside a local area network. Our framework utilises P2P techniques, which allows devices to function within the Internet with global scope in mind. The communication protocols used in UPnP are IP based and messages are sent between services using SOAP. Although these standards are open our framework abstracts the use of standards allowing interoperability between any open standards, not just IP and SOAP.

- *Service Discovery* – UPnP uses the Simple Service Discovery Protocol (SSDP) to discover services in the network. This is achieved by matching attribute-value pairs that allow pre-determined services such as printers and scanners to be discovered. The UPnP specification highlights that SSDP does not consider advanced querying. This is a major limitation of UPnP in that service descriptions and service requests must be pre-determined and in a format defined by the SSDP specification. If attribute-value pairs differ syntactically but mean the same thing semantically then service discovery fails. In our framework we provide a more advanced querying mechanism that allows service descriptions and service requests to be described using rich ontological structures. This significantly improves the matching process by allowing service descriptions and service requests to be matched not only at the syntactic level but at the semantic level as well. If the vocabularies are syntactically different but semantically equivalent our framework automatically resolves any terminology differences. This allows services to be more accurately matched within our framework than UPnP.

- *Device Capability Matching* – In UPnP devices provide a URL, which points to a UPnP description used to describe the device and the services it provides. When control points discover devices they use this URL to extract the description, which is then used to determine the devices capabilities. UPnP descriptions primarily focus on describing high-level information about the device and its services rather than the individual properties used to determine how resourceful the device is in terms of memory and processing power for example. Consequently it is difficult to use the UPnP standard to automatically determine what is the best device or service available within the network. In our framework we overcome this by using an adapted version of the CC/PP specification, which also uses our modified MAUT formula to provide an overall assessment of how well the device can execute the service it provides. This allows devices to select the best devices dependent on what is available within the network at any one time. This is a feature the UPnP specification does not provide.

- *Dynamic Service Composition* – There are no mechanisms within the UPnP specification to address dynamic service composition. Services are manually discovered and used via user interfaces. There are no mechanisms that allow devices to automatically discover ad hoc devices and services and compose them into high level compositions. In our framework we have addressed this limitation by providing semantic matching services that allow devices to query the network and form compositions, automatically with other devices and services within the network,

without any human intervention. Again this is a feature not supported in the UPnP specification.

- *Self-Adaptation* – There are no mechanisms within the UPnP specification to allow device configurations to be automatically composed or self-adapted based on environmental changes. Solutions are carefully choreographed and remain functional as long as all services in the solution remain operational. If a service fails then the whole solution may fail. In our framework services that provide the same functionality redundantly co-exist. If a service fails or a better service becomes available, device configurations are automatically adapted to ensure that the composition is maintained and that the best quality of service is provided. This marks a significant advantage over UPnP.

7.7.2 Open Services Gateway Initiative

- *Service-Oriented Networking* – OSGi is a service-oriented architecture, however the way services are hosted and served differs from our approach. OSGi service providers host services in the OSGi service container, which are controlled by service operators. These services can then be served via the internet to home networks using the OSGi gateway. This is an inherently centralised approach that provides services much like typical set-top box solutions in existence today. In our approach we have selected a less restrictive approach that utilises P2P technologies allowing for a greater number of services and increased flexibility to enable better and more innovative solutions. Any service within our framework can be used by any other device within the network without having to register with third-party registries. This allows services that provide the same functionality to redundantly co-exist and thus makes our framework far more flexible, scalable and fault-tolerant than OSGi.

- *Service Discovery* – OSGi provides service discovery mechanisms that allow services to be discovered that are contained in the OSGi Service Platform. Discovery is based on searching for services with pre-determined properties and a simple query language is used to select the required services needed. Again like UPnP services need to be described using predetermined vocabularies. As such discovering services that are syntactically distinct but semantically the same results in failure. In our framework we provide a more advanced service discovery mechanism than OSGi that allows devices to describe and discover services more accurately using high-level semantics. Furthermore devices discover services with global scope in mind using P2P technologies. We do not restrict services to proprietary service containers such as

OSGi, although our framework could accommodate this. This is a feature that OSGi does not support.

- *Device Capability Matching* – The OSGi specification (Version 3) does not address capability matching. Using services in OSGi is a manual process performed by the service provider, service operator and the user. We have argued that device compositions need to be created based on what devices and services within the network provide the best solution. In our framework services are provided that enable the device to determine how effectively the device can execute the service before it commits to using it. This is a feature not provided by OSGi. This feature is important for ubiquitous computing and services that reside within ad hoc environments such as P2P.

- *Dynamic Service Composition* – The OSGi specification does not provide any mechanisms to dynamically compose services without any human intervention. We have argued that managing device configurations is problematic and a better strategy is to develop mechanisms that allow devices themselves to do this. In our framework mechanisms allow devices to automatically discover and compose devices and services without any human intervention. This is a feature not supported by OSGi.

- *Self-Adaptation* – There are no self-adaptation mechanisms in OSGi. Service configurations are manually created and controlled. Like workflows service compositions remain operational as long as all services in the composition remain operational. Any faults that occur need to be manually corrected. In our framework any problems encountered within the composition are automatically corrected by discovering alternative services within the network and plugging them in without any human intervention. Again this is a feature not supported by OSGi.

7.7.3 Reconfigurable Ubiquitous Networked Embedded Systems

- *Service-Oriented Networking* – In RUNES device functions are abstracted as software services, which can be discovered and used within the network. This makes RUNES a service-oriented architecture that provides mechanisms to integrate services within the network. Services are plugged into RUNES using carefully created API interfaces. As such this is a proprietary protocol, much like USB, that provides a solution but ties device manufacturers into their protocol. It is not clear whether pre-defined interfaces can accommodate all device functionality. It is a question of granularity, which means that complex functions must be adapted to implement the interface methods provided by the RUNES API. In our framework we have tried to overcome this restriction using ontological structures to describe what services provide and how they can be

combined. In our framework devices propagate service requests that describe the data the candidate service must be capable of processing. Certain data may be defined as optional to make the matching process more flexible, as such our framework provides more scalable and flexible mechanisms to host and discover services that are not currently supported in RUNES.

- *Service Discovery* – The service discovery mechanism in RUNES, at present, is not clearly defined. They provide a generic interface method called *Advertisable*, which could support UPnP discovery. However restricting service discovery to the interface methods devices support is inflexible. It is based on pre-defined vocabularies that are syntactically matched. This solution will work in controlled environments, however applying the same service discovery technique within ad hoc networks that host heterogeneous devices is not possible. In our framework we overcome this limitation using flexible matching algorithms that are less restrictive than RUNES.

- *Device Capability Matching* – The RUNES specification does not define any mechanisms for selecting devices or services based on how capable they are or how effectively they can execute the services they provide. We have argued that in order to enable true ubiquity it is important to allow devices and services that provide the same functions to co-exist. As such mechanisms need to be provided that allow devices to decide what devices or services they use in order to create compositions that provide the solution. In our framework we have overcome this limitation and provided services that allow this to be achieved. Using these services devices can reason over what devices and services to include in final compositions based on how well they match the overall quality of service requirements.

- *Dynamic Service Composition* – RUNES supports dynamic service composition by allowing devices to discover advertisements containing pre-defined interfaces provided by the RUNES API. This is a restrictive form of dynamic service composition that works well in controlled environments but in true ubiquitous environments that are more ad hoc in nature, it would be difficult. In our framework we have foreseen this problem and provided better services capable of dynamically composing devices using rich ontological data. Devices can formulate semantic requests and propagate them within the network, which can be matched against the semantic descriptions of services. This makes our composition technique far more flexible, scalable and less restrictive than the approach adopted in RUNES. Consequently our framework can embrace ad hoc and infrastructure networks, which RUNES cannot.

- *Self-Adaptation* – The RUNES project supports self-adaptation. Through its carefully defined interfaces service compositions can detect and discover alternative services. As is the common theme with RUNES, self-adaptation is based on pre-determined interfaces, and as such it works well in controlled environments but not in ad hoc environments. Self-adaptation is closely interlinked with how devices and services are composed, and as such, restrictions in the higher levels filter through to the lower layers. Devices and services will be heterogeneous in nature and different middleware standards will be used. Consequently alternative mechanisms need to be developed that accommodate this uncertainty. Our framework has been developed with heterogeneity in mind and as such can self-adapt to changes between heterogeneous devices and services. This is something that RUNES cannot do.

7.8 Summary

Our framework has performed as expected and it has demonstrated that the challenges highlighted in Chapter 1 have been addressed. The overall performance of our prototype needs to be improved; however our primary focus was to demonstrate our ideas. This has been successfully achieved and provides a base solution on which to build.

Our evaluation shows that our framework surpasses current research initiatives within networked appliances and home networking and addresses a number of difficult problems. Many approaches adopt a human centric approach to interconnecting and managing networked appliances. We have argued that such models are inappropriate because it raises the question of who will perform these configuration and management tasks. It is becoming increasingly more difficult for home users and IT specialists alike to perform these tasks. Furthermore these approaches are too restrictive for innovative solutions. We have argued that alternative approaches are needed to automate this process. Our framework is such an approach and to the best of our knowledge is the first to address these issues within the field of networked appliances, which OSGi, UPnP and RUNES to name a few do not.

This Chapter was about evaluation, which grouped the key requirements of this thesis under five headings that our Networked Appliance Service Utilisation Framework must support. The opinions of key researchers have been quoted and linked to the requirements defined in Chapter 1. We have provided an evaluation of our framework and identified its strengths and weaknesses. This thesis presents a clear and viable design that allows networked appliances to exist within ad hoc networks and automatically discover semantically described services provided by other devices, based on device capability matching, which provides a basis for zero-configuration. It crosses several research disciplines and pulls together a number of key technologies such as networked appliances, home networking, P2P, ad hoc networking,

Semantic Web technologies and device capability matching. Where appropriate existing functionalities have been extended to include the secondary services provided by our framework.

Chapter 8

8 Conclusions and Future Work

8.1 Introduction

In this thesis we have stated that the proliferation of home appliances and the complex functions they provide make it ever harder for a specialist, let alone an ordinary home user, to configure and use them. To re-iterate the scenario described at the beginning of this thesis. Imagine your home environment, more specifically your living room, and the devices it contains. It is more than likely that it has a DVD player, VCR, Widescreen or Plasma TV, a surround sound speaker system, and a HiFi. Now imagine the time you bought your DVD Player and tried to integrate it with your existing device configuration. Like most people, you may have taken the DVD player out of the box and attempted to connect the wires to your TV and surround sound system and one hour later decided you needed to look at the instructions. After a further hour trying to understand the instructions, tuning in your TV and configuring your surround sound system you finally succeeded in viewing the DVD movie you bought. We have argued that these kinds of experiences are becoming increasingly more common and that it is no longer acceptable to burden the user, thus alternative mechanisms are required to abstract this complexity.

In this thesis we have focused on how to get different appliances, built to different specifications, to work together without having to change their original characteristics or protocols. Our research is about freeing users from the constraints imposed by physical machines. It's about breaking down machines and dispersing their operational capacity throughout our homes. Rather than severing ties between user and machines, we are actually forging a more intimate relationship between people and technology.

In trying to achieve this many challenges have had to be addressed, which include service-oriented networking; semantic service discovery; device capability matching; dynamic service composition, self-adaptation; and ubiquitous computing. We have argued that these challenges have been successfully addressed using our Networked Appliance Service Utilisation Framework. We have discussed in detail the core service-oriented middleware that comprises our framework, which integrates devices and the combined functions they provide. We have argued that our framework takes into account the capabilities devices support and

self-adapt and manage device configurations automatically. A case study is presented and a prototype solution has been developed that implements our framework.

In the remainder of this Chapter a summary of the thesis is presented, including the contributions made and the future work that needs to be carried out. This encompasses the difficulties encountered and the improvements required within the framework. This chapter is then concluded with final 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 middleware standards. 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 automatically form relationships, thus moving towards true zero-configuration. This chapter then briefly detailed a framework we developed that addresses these limitations enabling devices and services to be automatically integrated using flexible algorithms that perform the integration process using high-level semantic descriptions that describe the 'what' part of the composition rather than the 'how'. This chapter concluded by defining the scope of the research project, the novel contributions we have made and an outline of the thesis structure.

In Chapter 2 the background and related work was presented, which includes a discussion on the state of the art approaches within the field of Networked Appliances. This discussion defined the key concepts used within this thesis and described the limitations associated with current approaches and how they are addressed within this thesis. This chapter also discussed how networked appliances relate to home networking and described current middleware solutions that aim to seamlessly interconnect devices within home environments. A discussion was presented regarding how this integration is being performed using P2P techniques, where several P2P models where presented. Each P2P model was discussed in terms of their associated functions, merits and limitations and an argument was presented regarding how P2P techniques can be used to loosely connect devices within ad hoc network environments.

In this chapter we also looked at how techniques used within the Semantic Web could be used to address several limitations within current service-oriented middleware architectures, which included a discussion on service discovery and ontology evolution. The discussion argues that current service discovery mechanisms are inherently restrictive because they are based on proprietary descriptions that dictate how services must be described and discovered, which do not take into account the semantics or inherent vocabulary differences. As such an argument was presented pertaining to the use of semantics to better describe what services devices

provide and what they require. This Chapter also discussed current research relating to how ontologies can be used to describe services semantically, perform semantic interoperability and to dynamically compose devices and services.

A detailed discussion and the UML design models for our framework was presented in Chapter 3. This Chapter provides a high-level overview of our framework and briefly introduces the secondary services, which are discussed in more detail in Chapter 4. This Chapter then discusses the core service every device is required to implement, which allows the device to connect to the network.

Chapter 4 described in detail the UML design models for all the remaining secondary services that comprise our framework. These services allow devices to disperse their operational functions as independent services. They allow these services to be described and discovered using high-level semantics. These services also enable devices to determine how well a device is capable of executing a service it provides before committing to using it. They also manage device configurations and self-adapt when environmental changes are detected. We concluded this chapter by providing a summary and discussing what we have learnt and achieved during the design phase of this project.

In Chapter 5 an Intelligent Home Environment case study was presented which described how our framework could be used to automatically discover and compose devices and the services they provide, whilst at the same time providing the best quality of service. The case study also described how devices within the Intelligent Home Environment self-adapt based on environmental changes. Several other application scenarios where also presented indicating how our framework can be applied to other problem domains. We finally concluded this chapter with a summary and discussed what we have learnt from the case study and more importantly about our overall approach.

Chapter 6 presented a detailed discussion on how our framework was implemented. It discussed the toolsets used and highlighted the merits and shortcomings of several toolsets considered during the production of this thesis. It also presented the specifications that our framework conformed to and discussed how these specifications have been extended to realise our novel contributions. This included a detailed discussion about the technical details and explains how the prototype was developed. We concluded this chapter by providing a summary and discussing what we have learnt during the development of the prototype.

A qualitative evaluation of the NASUF implementation was presented in Chapter 7, which discussed the novel contributions the framework provides and how it was realised using our Intelligent Home Environment prototype system.

8.3 Contribution to knowledge

This thesis has presented a framework we have developed for integrating networked appliances within device and service-rich environments. The challenges we have overcome in order to achieve this include: service-oriented networking, semantic service discovery, dynamic service composition and device self-adaptive. We have addressed these challenges using our framework and made several novel contributions [Fergus 2003a, Fergus 2003b, Fergus 2003c, Fergus 2004, Mingkhwan 2004, Fergus 2005b, Fergus 2005a, Haggerty 2005, Mingkhwan 2005]. Our framework provides services that discover and interconnect devices within the network, enable operational functions to be discovered and composed using semantic matching, select devices based on the capabilities they support and mechanisms that allow device configurations to self-adapt to environmental changes. Each of these novel contributions is discussed in turn in the following subsections.

8.3.1 Service-Oriented Networking

In the area of service-oriented networking we have made several novel contributions, which we have published in [Fergus 2003a, Mingkhwan 2004, Fergus 2005a, Mingkhwan 2005]. Each contribution is listed below:

- Devices can dynamically integrate themselves within any environment and publish and dynamically discover services. Services may be pre-determined (middleware services that comprise our framework) as well as application specific (services wrapped around operational functions provided by devices) [Fergus 2003a], which can be simultaneously discovered and used by other devices within the environment [Mingkhwan 2004, Mingkhwan 2005].
- Our framework provides enhanced functions that allow devices and services within networked environments to be more accurately matched and integrated [Fergus 2005a].

8.3.2 Service Discovery

In the area of service discovery we have made several novel contributions, which we have also published in [Fergus 2003a, Fergus 2003b, Fergus 2003c, Fergus 2005a]. These contributions are listed below:

- Services are described and discovered based on their capabilities and mechanisms have been developed that perform better service matching than current attribute-value pair matching techniques – this allows devices to dynamically discover, compose and execute services based on peer collaborations, devoid of any human intervention [Fergus 2003a, Fergus 2005a].

- Service descriptions are serialised using high-level semantics that provide rich conceptual information about the individual functions devices provide [Fergus 2003b, Fergus 2003c].
- Device manufacturers are free to describe services using unconstrained vocabularies. Consequently, high-level semantics are used to resolve the inherent ambiguities between service requests and service descriptions [Fergus 2003b].
- Semantic service descriptions reside on individual devices and the total knowledge within the network is the sum of all devices and their associated semantic information. No centralised servers are used to store this information, thus semantic information is distributed within the network, which ensures flexibility, fault-tolerance and fair concept creation and evolution [Fergus 2003b].
- Semantic information is dynamically evolved devoid of any centralisation using general consensus. Concepts that are more commonly represented are emphasised whilst less common concepts are removed from the network over time. This is an automated process that requires no human intervention [Fergus 2003b].

8.3.3 Device Capability Matching

In the area of device capability matching we have also made several novel contributions, which have been published in [Mingkhwan 2004, Mingkhwan 2005]. These contributions are listed below:

- Devices and services that are similar in nature can redundantly co-exist within the framework and as such the same service can be provided by multiple devices. Device capabilities will vary so mechanisms have been developed that determine which device is better equipped to execute the given service [Mingkhwan 2004, Mingkhwan 2005].
- Existing device capability specifications have been extended to include capability scoring which not only assess individual device capabilities but also provide overall capability scores that assess the device as a whole. So even if a device is weak in one particular area, its overall capability score may still infer that it is the best device to use [Mingkhwan 2004, Mingkhwan 2005].

8.3.4 Dynamic service composition and self-adaptation

In the area of dynamic service composition we have made several novel contributions, which we have published in [Fergus 2005a]. Again each contribution is listed below:

- Devices can automatically form compositions with other devices to produce value added functions and aid zero-configuration [Fergus 2005a].

- Devices can self-adapt to environmental changes as and when devices or services become available or unavailable to ensure that device compositions are maintained [Fergus 2005a].
- Devices can form relationships with each other to create the best solution as specified in the capability models defined by each device. This ensures that the user's defined quality of service is either surpassed or maintained [Fergus 2005a].

8.3.5 Ubiquitous Computing

Lastly in the area of Ubiquitous Computing we have demonstrated that the new framework can be implemented on devices with limited capabilities. We have made several novel contributions, which again we have published in [Fergus 2004, Fergus 2005b].

- The framework is designed to work with devices with limited capabilities and has been implemented in a sensor network, which allows devices to be controlled using biofeedback [Fergus 2004]. Sensors connected to the body interact with sensors within the environment and when certain biological conditions are met, the devices are controlled [Bianchi 2003].
- The operational functions provided by devices are dispersed within networked environments using our framework, which harnesses the power of wireless and mobile technologies, thus reduce the wires and cables that are part and parcel of all modern day appliances [Fergus 2005b].

These novel contributions extend current advances in networked appliance and home networking research initiatives and provide a framework that is highly flexible, extensible and self-adaptive. Our framework moves us closer to seamlessly interconnecting devices and realising zero-configuration. Several open standards have been enhanced to provide additional functionality that surpasses the functions these standards describe. These extensions fit more efficiently within new and emerging intelligent network architectures to embrace ubiquitous and pervasive computing environments. Furthermore, our framework provides highly adaptive mechanisms that allow any device, irrespective of its capabilities, to function within the network and decide how the framework services are used.

Our evaluation demonstrates that our framework provides a viable solution. It highlights that using a distributed service-oriented architecture based on the peer-to-peer paradigm provides a better solution than existing workflow based approaches such as WSFL and BPEL4WS in that our framework allows numerous services, that provide the same functionality, to redundantly exist within the network – if a service fails an alternative service can be automatically discovered and used. This feature has been fully implemented and demonstrates that our framework is highly robust.

Our evaluation also shows that creating a single standard for describing and discovering services is highly unlikely. Device manufacturers and service providers will inherently use different vocabularies consequently our framework provides mechanisms capable of performing semantic mappings between vocabularies that are syntactically distinct but semantically equivalent. We show that using ontologies aids this mapping process and provides a highly flexible mechanism for service discovery that can accommodate a broader range of queries that surpasses existing service discovery mechanisms currently being using in frameworks such as OSGi, UPnP and DLNA.

Again utilising peer-to-peer networking our implementation has demonstrated how devices are treated as individual knowledge nodes that can share and evolve ontological structures using our framework services. Devices contain their own knowledge used to describe the services they provide, which can be shared with other devices within the network. We have shown that these knowledge structures can be evolved over time using peer-to-peer techniques and aid semantic interoperability between different vocabularies.

Our evaluation argues that typically compositions are inherently human centric and as such the overall quality of device configurations is determined by the user. In our framework device configurations and compositions are automated, as such our framework provides a service that selects devices that provides the best quality of service.

We have found that dynamically composing devices and services is a difficult problem and as such most research initiatives base their solutions on manual or semi-automated techniques. In our framework however we have tried to address this challenge using techniques used within the Semantic Web. Our evaluation illustrates that our matching process is more flexible than existing approaches, such as OSGi, UPnP and DLNA and provides a base solution.

Little work within the area of networked appliances and home network has been done in the area of self-adaptation and as we have argued in this thesis it is becoming increasing more difficult to manage device configurations. Our framework aims to relieve the user from the management tasks associated with interconnecting devices using a self-adaptation service that allows devices to form relationships with devices in the network as soon as they have been switched on and self-adapt to any environment changes that may occur.

8.4 Further Work

The implementation and case study evaluation demonstrate that a contribution to knowledge has been made and that the research carried out addresses several research problems. However, our work has also encountered difficulties along the way and has raised a number

of interesting questions. As such this section provides details of the questions raised, which are the subject of future research within the Networked Appliances Laboratory at Liverpool John Moores University.

8.4.1 Semantic Annotation and Processing Issues

We still need to look at the co-existence of correct and incorrect information within device ontologies. We do not make any assumptions that the information created by device manufacturers will be correct or consistently represented in a pre-determined knowledge structure. Consequently different device manufacturers will classify concepts differently and in some cases incorrectly. Furthermore the ontology evolution process is problematic and time consuming. The semantic matching algorithm needs to be optimised in order to speed up the evolutionary process.

8.4.2 Security

One of the key functions that NASUF does not address is that of security. Ad hoc environments raise an important question regarding trustworthiness of service providers. The middleware must ensure that the content received from services is authenticated and that data streams are not intercepted and altered during transmission. In this way, trust between network entities may be maintained. This becomes an important requirement within ad hoc environments, which resist any form of centralised control. To address this challenge a lightweight trust mechanism needs to be developed which guarantees the data transferred between services has not been altered or redirected during transmission and which accommodates different levels of integrity dependent on what type of data is being transferred. For example transmitting payment details or documents between devices will require that the highest level of integrity is maintained by encrypting every packet that is sent, whilst streaming multimedia data may require less integrity where only every 100^{th} packet needs to be encrypted. The trust mechanisms must also be lightweight and capable of being installed on any device irrespective of its capabilities.

In future work a lightweight mechanism for maintaining trust in ad hoc multimedia networks will be developed, which will prevent the modification of data in transit. Development will ensure that the computational overhead incurred by the posited scheme is minimal, whilst ensuring that the content received by devices is free from modification. Our contribution will be to extend existing authentication mechanisms to ensure trust is continually maintained whilst data streams are being transmitted between different services [Haggerty 2005].

8.4.3 Feature Interaction

Another key requirement that needs to be addressed relates to Feature Interactions. Services may operate well when used in isolation or within small compositions, however problems will occur when trying to interwork a large number of services at the same time. A body of work is currently being carried out elsewhere [Kolberg 2002, Kolberg 2003] to address the challenges associated with Feature Interactions and it is envisaged that these research efforts could be integrated into NASUF.

8.4.4 Service and Device Composition Issues

The concrete matching algorithm needs to be fully implemented. This will allow high-level semantics to be mapped to low-level service interfaces. In the prototype we demonstrate this using a simple test case, however a more complex mechanism is required. Work has begun within this area [Fergus 2005a], however further research is required before this feature can be fully functional within NASUF.

8.4.5 Transport Protocol Interoperability

At present the NASUF implementation is mapped onto the TCP/IP protocol because it is the most common networking protocol currently used. However in the future interoperability between different transport protocols will be investigated. The P2P implementation used within NASUF is JXTA and at present the Java and C bindings only consider TCP/IP. The documentation states that future bindings will be developed, consequently a future project will interlink with current interoperability research being carried out [Abuelma'atti 2002a, Abuelma'atti 2002b] to bridge between different wireless technologies such as 802.11x, Bluetooth and RF. This research will look at creating software adaptors as services that can be dynamically discovered and integrated within NASUF. These adapters will automatically form bindings with JXTA so as to maintain a unified addressing scheme.

8.5 Concluding Remarks

We are currently seeing the convergence of several key technologies whereby devices are becoming more interconnected. Advances in global and wireless communications have opened up the possibility for new and novel solutions that are changing the way we use and interact with the devices we own. User demands and these technological advances are moving us closer to the pervasive computing vision. The home of the future will include networked appliances that disperse their operational functions as middleware services providing flexible, intuitive and zero-maintenance mechanisms for dynamic service composition, deployment, extensibility, management and usage. Whilst much work exists relating to service-oriented

frameworks, this typically relies on attribute-based service matching and discovery, which is inherently restrictive since no universally agreed service description or taxonomy is available to describe services homogeneously. Device manufacturers inadvertently use different vocabularies to describe services and therefore ambiguities between terminologies are likely.

In this thesis these requirements have been successfully addressed by designing and developing a new framework called NASUF. This framework allows services to be described using machine-processable semantics. This enables devices to make informed decisions regarding service compositions. The framework is self-adaptive and is capable of resolving device or service failures within compositions as and when they occur.

Through peer collaborations devices successfully form dynamic compositions with other devices and the services they provide by processing ontological contextual descriptions, which guide the composition process. These descriptions describe the high-level concepts that relate to the "what" part of the service composition rather than the "how". Consequently each device provides ontological descriptions, which are dynamically evolved over time. Services are composed based on the low-level signatures each service provides devoid of any human intervention, which are not known beforehand. Mechanisms to achieve this give rise to the true potential of service-oriented architectures by creating value-added services, whereby global functionality cannot be produced by one single device or service alone. As such this framework successfully provides mechanisms that allow services to be composed based on the semantic similarities between the capabilities they support. High-level semantic descriptions are developed and mapped onto the syntactic signatures used to describe services, which form explicit mappings between the inputs one service requires and the outputs another service produces.

Our framework successfully incorporates devices of varied capabilities using mechanisms that perform capability matching. Before services provided by devices are composed, the framework determines if the device providing the service has the required hardware, software and networking capabilities to effectively execute it.

In this thesis we have provided a detailed overview of the background and related work and discussed the influential factors. Developing our framework has been multi-disciplinary which has utilised and extended existing research initiatives and open standards to produce a flexible open middleware architecture that allows devices and services to be seamlessly interconnected, which abstracts the underlying implementation details. Thus this thesis provides a broad platform on which to integrate next generation networked appliances.

We have successfully illustrated that we have made several novel contributions that extend far beyond existing networked appliance and home networking architectures. We have allowed

devices to redundantly disperse framework and operational functions within the network; we have allowed services to be more accurately discovered using high-level semantics; we have allowed devices to automatically select devices that best execute the services they provide; and we have provided mechanisms to manage device configurations and allowed them to automatically self-adapt to any environmental changes that occur.

We have successfully implemented our framework and produced a prototype that successfully demonstrates how our framework services work. The prototype implements our case study, which is an intelligent home environment, and illustrates how individual functions provided by devices can be dispersed 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 (a full list can be founding Appendix E). We have published papers on how to embed device functions within the network as individual services; semantic discovery mechanisms; device capability matching; and self-adaptation.

Our framework is designed to reduce costs. Currently we are required to upgrade and replace devices to support new and emerging standards even though a large percentage of the functions provided by new and old devices alike remain the same. Our framework allows devices to evolve over time to include new functions that they where not initially designed to do. For example a DVD player can automatically download a required codec when it is presented with a media format it cannot process. Devices to date do not work in this way – if you wish to use a multimedia format your player does not support then you have to buy a new player. This is inefficient and costly to the consumer. Using our framework, conflicts like this can be automatically detected and rectified. No other framework provides this functionality.

Overall this has been a successful project and has generated a lot of interest. It has allowed us to explore how technological advances will progress and we believe that we are ahead of current solutions. Although it is difficult to predict how technology will change it is clear that networked appliances and home networking is becoming more sophisticated and it is reasonable to say that IT will play a major role in how it is managed. Our framework provides a viable solution that can be used to reduce the inherent complexity this will bring and automatically mange the device configuration and management tasks.

REFERENCES

[Aberer 2003] Aberer, K., Cudre-Mauroux, P., Hauswirth, M., "The Chatty Web: Emergent Semantics Through Gossiping," In Proceedings of *The 12th International World Wide Web Conference*, pp.197-206, Budapest, Hungary, Springer, (May 2003).

[Abuelma'atti 2002a] Abuelma'atti, O., Merabti, M. Askwith, B., "Interworking the Wireless Domain," In Proceedings of *Third International Symposium in Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pp.344-347, Staffordshire, UK, (July 2002).

[Abuelma'atti 2002b] Abuelma'atti, O., Merabti, M. and Askwith, B., "A Wireless Networked Appliances MAC Bridge," In Proceedings of *5th IEEE International Workshop on Networked Appliances*, pp.96-101, Liverpool, IEEE Computer Society, (October 2002).

[OSGi Alliance 2005] OSGi Alliance, "The OSGi Service Platform - Dynamic services for networked devices," http://www.osgi.org/, (Accessed: 2006).

[Andrews 2005] Andrews, T., *et al.*, "Business Process Execution Language for Web Services Version 1.1," http://www-128.ibm.com/developerworks/library/specification/ws-bpel/, (Accessed: 2006).

[Arora 2003] Arora, A. C., Pabla, K. S., "JXTA for J2ME (JXME) Project," http://jxme.jxta.org/, (Accessed: 2006).

[IEEE Standards Association 2005] IEEE Standards Association, "IEEE Standards Association," http://standards.ieee.org/, (Accessed: 2006).

[Barba 2005] Barba, A., "HomeOnAir: WAP Access for a Home Automation Server," http://icadc.cordis.lu/fep-cgi/srchidadb?ACTION=D&CALLER=PROJ_IST&QF_EP_RPG=IST-1999-20138, (Accessed: 2006).

[BBN 2004] BBN, "The ARPANET: Forerunner of Today's Internet," http://www.bbn.com/Historical_Highlights/?name=Arpanet.html&search=arpanet, (Accessed: 2006).

[Berners-Lee 1989] Berners-Lee, T., "Information Management: A Proposal," http://www.w3.org/History/1989/proposal.html, (Accessed: 2006).

[Berners-Lee 1998] Berners-Lee, T., "What the Semantic Web can represent," http://www.w3.org/DesignIssues/RDFnot.html, (Accessed: 2006).

[Berners-Lee 2000] Berners-Lee, T., "Weaving The Web," Texere Publishing Limited, London, ISBN: 1-58799-018-0, (2000).

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

[BETSY 2005] BETSY, "BEing on Time Saves energY: Continuous Multimedia Experiences on Networked Handheld Devices," http://www.extra.research.philips.com/euprojects/betsy/index.htm, (Accessed: 2006).

[Bhatti 2002] Bhatti, G., Sahinoglu, Z., Peker, K. A., Guo, J. and Matsubara, F., "A TV-Centric Home Network to Provide a Unified Access to UPnP and PLC Domains," In Proceedings of *IEEE Fourth International Workshop on Networked Appliances (IWNA)*, pp.234-242, Gaithersburg, USA, IEEE Computer Society, (January 2002).

[Bianchi 2003] Bianchi, L., Babiloni, F., Cincotti, F., Arrivas, M., Bollero, P. and Marciani, M. G., "Developing wearable bio-feedback systems: a general-purpose platform," *IEEE Neural Systems and Rehabilitation Engineering*, vol.11 (2), pp.1-3, (2003).

[Brandenburg 1999] Brandenburg, K., "MP3 and ACC Explained," In Proceedings of *The Proceedings of the AES 17th International Conference on High-Quality Audio Coding*, Florence, Italy, (September 1999).

[Brooks 2002] Brooks, R. A., "Robot: The future of flesh and machines," Penguin Books, London, ISBN: 0140297189, (2002).

[Burk 1998] Burk, R., "Unix Unleashed," Sams Publishing, Indianapolis, Indiana, USA, Third Edition, ISBN: 0672314118, (1998).

[CEA 2000] CEA, "VHN Home Network Specification," EIA/CEA-851, Consumer Electronics Association, (2000).

[CEBus 2005] CEBus, "Bringing Interoperability to Home Networks," http://www.cebus.org, (Accessed: 2006).

[CEPCA 2005] CEPCA, "Consumer Electronics Powerline Communication Alliance (CEPCA)," http://www.cepca.org/home, (Accessed: 2006).

[Chakraborty 2003] Chakraborty, D. and Joshi, A., "Anamika: Distributed Service Composition Architecture for Pervasive Environments," In Proceedings of *The Ninth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp.38-40, San Diego, California, USA, ACM Press, (September 2003).

[Chemishkian 2002] Chemishkian, S., "Building smart services for Smart Home," In Proceedings of *IEEE 4th International Workshop on Networked Appliances (IWNA)*, pp.215-224, Gaithersburg, Maryland, USA, IEEE Computer Society, (January 2002).

[Chen 2003] Chen, L., Schadbolt, N. R., Goble, C., Tao, F., Cox, S. J., Puleston, C. and Smart, P., "Towards a Knowledge-based Approach to Semantic Service Composition," In Proceedings of *2nd International Semantic Web Conference (ISWC2003)*, pp.319-334, Florida, USA, Springer, (October 2003).

[Cheng 2000] Cheng, L. and Marsic, I., "Lightweight Service Discovery for Mobile AdHoc Networked Appliances," In Proceedings of *Second International Workshop on Networked Appliances*, Rutgers University, New Jersey, USA.

[Chen-Mie 1995] Chen-Mie, W., Dah-Jyh, P., Wen-Tsung, C. and Jian-Shing, H., "A high-performance system for real-time video image compression applications," *IEEE Transactions on Consumer Electronics*, vol.41 (1), pp.125 - 131.

[CyCorp 2002] Cycorp, "Writing Efficient CycL: Some Concrete Suggestions," Web Site, OpenCyc.org, http://www.opencyc.org/doc/tut/?expand_all=1, (Accessed: 17-11-2005).

[Dabek 2001] Dabek, F., Brunskill, E., Kaashoek, F. and Karger, D., "Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service," In Proceedings of *The Eighth Workshop on Hot Topics in Operating Systems*, pp.81-86, Germany, IEEE Computer Society, (May 2001).

[Daconta 2003] Daconta, M. C., Obrst, L. J. and Smith, K. T., "The Semantic Web - A Guide to the Future of XML, Web Services, and Knowledge Management," Wiley Publishing Inc., Indianapolis, Indiana, ISBN: 0471432571, (2003).

[DAML 2003a] DAML, "DAML+OIL," Web Site, http://www.daml.org, (Accessed: 2006).

[DAML 2003b] DAML, "DAML-S: Semantic Markup for Web Services," Web Site, DAML Org, http://www.daml.org/services/daml-s/0.9/daml-s.pdf, (Accessed: 17-11-2005).

[DAML 2003c] DAML, "OWL-S 1.0 Release," http://www.daml.org/services/owl-s/1.0/, (Accessed: 2006).

[DARPA 2003] DARPA, "Defence Advanced Research Projects Agency," http://www.darpa.mil/, (Accessed: 2006).

[Dean 2005] Dean, T., "Network+ 2005 in Depth," Thomson Course Technology PTR, Boston, USA, ISBN: 1592007929, (2005).

[Decker 2000] Decker, S., Melnik, S., van Harmelen, V., Fensel, D., Klein, M., Broekstra, J., Erdmann, M. and Horrocks, I., "The Semantic Web: The Roles of XML and RDF," *IEEE Internet Computing*, vol.4 (5), pp.63-74, (September/October 2000).

[DHWG 2003] DHWG, "Digital Home White Paper," http://www.dhwg.org/resources/DHWG_WhitePaper.pdf, (June 2003).

[Dimitrov 2000] Dimitrov, M., "XML Standards for Ontology Exchange," In Proceedings of *Proceedings of OntoLex 2000: Ontologies and Lexical Knowledge Bases*, pp.153-188, Sozopol, Bulgaria, (September 2000).

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

[Douglas 2004] Douglas, S. and Douglas, K., "Linux Timesaving Techniques for Dummies," John Wiley & Sons Publishing, New York, USA, ISBN: 0764571737, (2004).

[Dutta-Roy 1999] Dutta-Roy, A., "Networks for Homes," *IEEE Spectrum*, vol.36 (12), pp.26-33, (December 1999).

[Eberspacher 2004] Eberspacher, J., Schollmeier, R., Zols, S. and Kunzmann, G., "Structured P2P Networks in Mobile and Fixed Environments," In Proceedings of *2nd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, West Yorkshire, UK, (July 2004).

[Evans 2001] Evans, D., "In-home wireless networking: an entertainment perspective," *IEEE Electronic and Communication Engineering*, vol.13 (5), pp.213-219, (October 2001).

[Farquhar 1997] Farquhar, A., Fikes, R. and Rice, J., "The Ontolingua Server: a tool for collaborative ontology construction," *International Journal of Human-Computer Studies*, vol.46 (6), pp.707-727, (June 1997).

[Feibel 2000] Feibel, W., "The Network Press Encyclopaedia of Networking," Sybex, London, 3rd Edition, ISBN: 0782122558.

[Fensel 2001] Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L. and Patel-Schneider, P. F., "OIL: An Ontology Infrastructure for the Semantic Web," *IEEE Intelligent Systems*, vol.16 (2), pp.38-45, (March/April 2001).

[Fensel 2002] Fensel, D., Staab, S., Studer, R. and van Harmelen, F., "Towards the Semantic Web: Ontology-driven Knowledge Management," John Wiley & Sons, Ltd, Chichester, West Sussex, England, ISBN: 0470848677, (2002).

[Fensel 2003] Fensel, D., Hendler, J., Lieberman, H. and Wahlster, W., "Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential," MIT Press, London, England, ISBN: 0262062321, (2003).

[Fergus 2003a] Fergus, P., Mingkhwan, A., Merabti, M. and Hanneghan, M., "DiSUS: Mobile Ad Hoc Network Unstructured Services," In Proceedings of *(PWC'2003) Personal Wireless Communications*, pp.484-491, Venice, Italy, Springer, (September 2003).

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

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

[Fergus 2004] Fergus, P., Merabti, M., Hanneghan, M. B. and Taleb-Bendiab, A., "Controlling Networked Devices in Ubiquitous Computing Environments using Biofeedback," In Proceedings of *The 5th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting*, pp.91-96, Liverpool, UK, John Moores University, (June 2004).

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

[Fergus 2005b] Fergus, P., "Welcome to the Wireless Age," In Proceedings of *Review*, pp.34 - 35, Liverpool John Moores University, (November).

[Fujii 2004] Fujii, K. and Suda, T., "Dynamic Service Composition Using Semantic Information," In Proceedings of *2nd International Conference on Service Oriented Computing*, pp.39-48, NY, USA, ACM Press, (November 2004).

[Genesereth 1991] Genesereth, M. R., "Knowledge Interchange Format," In Proceedings of *2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp.599-600, Cambridge, Massachusetts, USA, (April 1991).

[Gillett 2000] Gillett, S. E., Lehr, W. H., Wroclawski, J. T. and Clark, D., "A Taxonomy of Internet Appliances," In Proceedings of *(TPRC2000) Telecommunications Policy Research Conference*, Alexandria, VA, USA, (September 2000).

[Gillett 2001] Gillett, S. E., Lehr, W. H., Wroclawski, J. T. and Clark, D., "Do Appliances Threaten Internet Innovation?," *IEEE Communications Magazine*, vol.39 (10), pp.46-51, (October 2001).

[Gnutella 2001] Gnutella, "The Gnutella Protocol Specification v0.4," http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf, (Accessed: 2006).

[Goldfarb 2002] Goldfarb, C. F. and Prescod, P., "XML Handbook," Prentice Hill PTR, Upper Saddle River, NJ 07458, 4th Edition, ISBN: 0130651982, (2002).

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

[Gradecki 2002] Gradecki, J. D., "Mastering JXTA: Building Java Peer-to-Peer Applications," Wiley Publishing, Inc., Indianapolis, Indiana, USA, ISBN: 0471250848, (2002).

[Le Grand 2001] Le Grand, B., Soto, M. and Dodds, D., "XML Topic Maps and Semantic Web Mining," In Proceedings of *12th European Conference on Machine Learning/5th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD -2001)*, Freiburg, Germany, (September 2001).

[Grokster 2005] Grokster, "Grokster," http://www.grokster.com/, (Accessed: 2006).

[Gruber 1993] Gruber, T. R., "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol.5 (2), pp.199-220, (June 1993).

[Haarslev 2001] Haarslev, V. and Moller, R., "Description of the RACER System and its Applications," In Proceedings of *International Workshop on description Logics*, pp.131-141, Stanford, USA, (August 2001).

[Haggerty 2005] Haggerty, J., Shi, Q., Fergus, P. and Merabti, M., "Data Authentication and Trust within Distributed Intrusion Detection System Inter-Component Communications," In Proceedings of *(EC2ND'05) 1st European Conference on Computer Network Defence*, pp.197-206, University of Glamorgan, UK, Springer, (December 2005).

[Halepovic 2002] Halepovic, E. and Deters, R., "Building a P2P Forum System with JXTA," In Proceedings of *Second International Conference on Peer-to-Peer Computing (P2P'02)*, pp.41-48, Linkoping, Sweden, IEEE Computer Society, (September 2002).

[HAVI 2003] HAVI, "HAVI, the A/V digital network revolution," San Mamon, CA, http://www.havi.org/pdf/white.pdf, (2003).

[Heflin 1998] Heflin, J., "Semantic Search - The SHOE Search Engine," Web Site, http://www.cs.umd.edu, http://www.cs.umd.edu/projects/plus/SHOE/search/, (Accessed: 04-08).

[Heflin 2000] Heflin, J. and Hendler, J., "Dynamic Ontologies on the Web," In Proceedings of *Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pp.443-449, Austin, Texas, U.S.A., AAAI/MIT, (July 2000).

[Heflin 2003] Heflin, J. and Huhns, M. N., "The Zen of the Web," *IEEE Internet Computing*, vol.7 (5), pp.30-33, (September/October 2003).

[HES 2005] HES-Standards, "Home Electronic System Standards," ISO/IEC JTC 1/SC 25/WG 1, www.hes-standards.org, (Accessed: 2005).

[Hightower 2002] Hightower, R. and Lesiecki, N., "Java Tools for Extreme Programming: Mastering Open Source Tools Including Ant, JUnit, and Cactus," John Wiley & Sons, Inc., New York, ISBN: 0-471-20708-X, (2002).

[Future Home 2005] Future Home, "The Future Home Project," http://future-home.org, (Accessed: 2006).

[HomeTalk 2005] HomeTalk, "The HomeTalk Project," Web Site, http://www.hometalk.org/, (Accessed: 2006).

[Horridge 2004] Horridge, M., Knublauch, H., Rector, A., Stevens, R. and Wroe, C., "A Practical Guide To Building OWL Ontologies Using the Protege-OWL Plugin and CO-ODE Tools Edition 1.0," Tutorial, University of Manchester, University of Stanford, Manchester, http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf, (August 2004).

[Horrocks 2005] Horrocks, I., "The FaCT System," http://www.cs.man.ac.uk/~horrocks/FaCT/, (Accessed: 2006).

[HP Labs 2004] HP Labs., "Jena - A Semantic Web Framework for Java," http://jena.sourceforge.net, (Accessed: 2006).

[IETF 2004] IETF, "Session Initiation Protocol (SIP)," http://www.ietf.org/html.charters/sip-charter.html, (Accessed: 2006).

[iMesh Inc 2005] iMesh Inc., "iMesh Professional 5.0," http://www.imesh.com/, (Accessed: 2006).

[Sun Microsystems Inc. 2005a] Sun Microsystems Inc., "JXTA v2.3.x: Java Programmer's Guide," http://www.jxta.org/docs/JxtaProgGuide_v2.3.pdf, (Accessed: 2006).

[Sun Microsystems Inc. 2005b] Sun Microsystems Inc., "Java Media Framework," http://java.sun.com/products/java-media/jmf/, (Accessed: 2006).

[Sun Microsystems Inc. 2005c] Sun Microsystems Inc., "JXTA 2.3.1," http://www.jxta.org/, (Accessed: 2006).

[Intel 2003] Intel, "Intel Digital Homes," http://www.intel.com/technology/digitalhome/, (Accessed: 2006).

[ISO/IEC 2001] ISO/IEC, "Interconnection of Information Technology Equipment: Home Electronic System," ISO/IEC JTC 1/SC 25/WG 1, (2001).

[Jacob 2004] Jacob, M., "RDF in the Semantic Hifi European project," In Proceedings of *1st Italian Workshop on Semantic Web Applications and Perspectives (SWAP)*, pp.50-54, Ancona, Italy, (December 2004).

[JXTA 2001] JXTA, "Project JXTA: An Open, Innovative Collaboration," http://www.jxta.org/project/www/docs/OpenInnovative.pdf, (2001).

[Karp 2005] Karp, P. D., Chaudhri, V. K. and Thomere, J., "XOL: XML-Based Ontology Language," http://www.ai.sri.com/pkarp/xol/, (Accessed: 2006).

[Kent 2005] Kent, R. E., "Ontology Markup Language," http://www.ontologos.org/OML/OML%200.3.htm, (Accessed: 2006).

[Klyne 2004] Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. H. and Tran, L., "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0," http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/, (Accessed: 2006).

[Kolberg 2002] Kolberg, M., Magill, E., Marples, D. and Tsang, S., "Feature Interactions in Services for Internet Personal Appliances," In Proceedings of *(ICC'02) IEEE International Conference on Communications*, pp.2613-2618, New York, USA, IEEE Computer Society, (April 2002).

[Kolberg 2003] Kolberg, M., Magill, E. H. and Wilson, M., "Compatibility Issues between Services Supporting Networked Appliances," *IEEE Communications Magazine*, vol.41 (11), pp.136 - 147, (November 2003).

[Koumpis 2005] Koumpis, C., Hanna, L., Anderson, M. and Johansson, M., "Beyond wireless cable-replacement for industrial control and monitoring: The RUNES approach," In Proceedings of *Profibus International Conference*, Warwickshire, UK, (June 2005).

[Kumar 2003] Kumar, R., Poladian, V., Geenberg, I., Messer, A. and Milojicic, D., "Selecting Devices for Aggregation," In Proceedings of *(WMCSA'03) Fifth IEEE Workshop on Mobile Computing Systems and Applications*, pp.150-159, Monterey, California, USA, IEEE Computer Society, (October 2003).

[Langton 1996] Langton, C. G., "Artificial Life," "The Philosophy of Artificial Life." Edited by M. A. Boden. New York, Oxford University Press Inc.**:** 39-93.

[Lea 2000] Lea, R., Gibbs, S., Dara-Abrams, A. and Eytchison, E., "Networking home entertainment devices with HAVi," *IEEE Computer*, vol.33 (9), pp.35 - 43, (2000).

[Lee 2002] Lee, S. and Smelser, T., "Jabber Programming," M&T Books, Hungry Minds Inc.,, 909 Third Avenue, New York, NY 10022, ISBN: 0-7645-4934-0, (2002).

[Leymann 2004] Leymann, F., "Web Services Flow Language (WSFL) Version 1.0," http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, (Accessed: 2006).

[Liu 2004] Liu, J. and Issarny, V., "QoS-Aware Service Location in Mobile Ad-Hoc Networks," In Proceedings of *(MDM'04) IEEE International Conference on Mobile Data Management*, pp.224-235, Berkeley, California, USA, IEEE Computer Society, (January 2004).

[Lime Wire LLC 2005] Lime Wire LLC, "LimeWire," http://www.limewire.com/english/content/home.shtml, (Accessed: 2006).

[Madhusudan 2004] Madhusudan, T. and Uttamisingh, N., "A declarative approach to composing web services in dynamic environments," *Decision Support Systems*, vol.In Press.

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

[Marples 2001] Marples, D. and Kriens, P., "The Open Services Gateway Initiative: An Introductory Overview," *IEEE Communications Magazine*, vol.39 (12), pp.110-114, (December 2001).

[Marshall 2001] Marshall, P., "Home networking: a TV perspective," *IEEE Electronic and Communication Engineering*, vol.13 (5), pp.209-212, (October 2001).

[McGuinness 2000] McGuinness, D. L., Fikes, R., Rice, J. and Wilder, S., "An Environment for Merging and Testing Large Ontologies," In Proceedings of *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and*

Reasoning (KR2000), pp.483-493, Breckenridge, Colarado, USA, Morgan Kaufmann Pubishers, (April 2000).

[McGuinness 2001] McGuinness, D. L., "Ontologies Come of Age," "The Semantic Web: Why, What, How." Edited by. London, England, MIT Press.

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

[McIlraith 2003] McIlraith, S. A. and Martin, D. L., "Bringing semantics to Web services," *IEEE Intelligent Systems*, vol.18 (1), pp.90-93, (Jannuary/February 2003).

[Medjahed 2003] Medjahed, B., Bouguettaya, A. and Elmagarmid, A. K., "Composing Web Services on the Semantic Web," *The International Journal of Very Large Data Bases*, vol.12 (4), pp.333-351, (November 2003).

[Meessen 2004] Meessen, J., Parisot, C., Lebarz, C., Delaigle, J. F. and Nicholson, D., "IST WCAM Project : Smart and Secure Video Coding Based on Content Detection," In Proceedings of *European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology*, London, U.K., (November 2004).

[Microsoft Corp. 2003] Microsoft Corp., "UPnP Device Architecture 1.0: Service Description," http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf, (December 2003).

[Microsoft Corp. 2005] Microsoft Corp., "UPnP Forum," http://www.upnp.org/, (Accessed: 2006).

[Milanovic 2004] Milanovic, M. and Malek, M., "Current Solutions for Web Service Composition," *Internet Computing*, vol.80 (6), pp.51-59, (Nov/Dec 2004).

[Millar 2004] Millar, W., Collingridge, R. J. and Ward, D. A., "Consumer vehicle telematics - an emerging market where web Services offer benefits," *BT Technology Journal*, vol.22 (1), pp.99-106, (March 2004).

[Miller 2001] Miller, B., Nixon, T., Tai, C. and Wood, M. D., "Home Networking with Universal Plug and Play," *IEEE Communications Magazine*, vol.39 (12), pp.104-109, (2001).

[Mingkhwan 2002] Mingkhwan, A., Merabti, M. and Askwith, B., "Interoperability of Structured and Unstructured Services in Personal Mobility Information Space," In Proceedings of *European Wireless 2002*, Florence, Italy, (2002).

[Mingkhwan 2003] Mingkhwan, A., Merabti, M., Askwith, B. and Hanneghan, M., "Global Wireless Framework," In Proceedings of *European Personal Mobile Communications Conference (EPMCC'03)*, Glasgow, Scotland, (2003).

[Mingkhwan 2004] Mingkhwan, A., Fergus, P., Abuelma'atti, O. and Merabti, M., "Implicit Functionality: Dynamic Services Composition for Home Networked Appliances," In

Proceedings of *(ICC'2004) IEEE International Conference on Communications*, pp.43-47, Paris, France, IEEE Computer Society, (June 2004).

[Mingkhwan 2005] Mingkhwan, A., Fergus, P., Abuelma'atti, O., Merabti, M., Askwith, B. and Hanneghan, M., "Dynamic Service Composition in Home Appliance Networks," *(MTAP) Multimedia Tools and Applications: A Special Issue on Advances in Consumer Communications and Networking*, vol.31 (1), (December 2006).

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

[Mitra 2000] Mitra, P., Wiederhold, G. and Kersten, M. L., "A Graph-Oriented Model for Artriculation of Ontology Interdependencies," In Proceedings of *7th International Conference on Extending Database Technology*, pp.80-100, Konstanz, Germany, Springer, (March 2000).

[Morle 2003] Morle, P., Morris, A. and Hemming, N., "KaZaa," http://www.zeropaid.com/kazaalite/, (Accessed: 2006).

[Moyer 2000] Moyer, S., Marples, D., Tsang, S. and Ghosh, A., "Service Portability of Networked Appliances," *IEEE Communications Magazine*, vol.40 (1), pp.116-121, (January 2000).

[Murhammer 1998] Murhammer, W., et al., "TCP/IP Tutorial and Technical Overview," IBM Redbooks, Research Triangle Park, NC, 27709-2195, 6th Edition, ISBN: 0738412007.

[Naisbitt 1991] Naisbitt, J. and Aburdene, P., "Megatrends 2000: New directions for tomorrow," Avon Books, Inc., New York, ISBN: 0380704374, (1991).

[Narayanan 2002] Narayanan, S. and McIlraith, S. A., "Simulation, Verification and Automated Composition of Web Services," In Proceedings of *Proceedings of the eleventh international conference on World Wide Web*, pp.77-88, Honolulu, Hawaii, USA, ACM Press, (May 2002).

[StreamCast Networks 2005] StreamCast Networks, "Morpheus," http://morpheus.com/, (Accessed: 2005).

[Nikolova 2003] Nikolova, M., Meijs, F. and Voorwinden, P., "Remote mobile control of home appliances," *IEEE Transactions on Consumer Electronics*, vol.49 (1), pp.123 - 127, (2003).

[Noy 2000] Noy, N. F. and Musen, M. A., "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," In Proceedings of *The Seventeenth National Conference on Artificial Intelligence (AAAI'00)*, pp.450-455, Austin, Texas, USA., AAAI Press/The MIT Press, (July 2000).

[Oaks 2002] Oaks, S., Traversat, B. and Gong, L., "JXTA in a Nutshell," O'Reilly Associates, ISBN: 0-596-00236-x, (2002).

[Oasis 2005] Oasis, "UDDI," http://www.uddi.org, (Accessed: 2006).

[Oram 2001] Oram, A., Minar, N. and Hedlund, M., "Peer-to-Peer: Harnessing the Power of Disruptive Technologies," O'Reilly, 1005, Gravenstein Highway North, Sebastopol, CA 95472, ISBN: 0-596-00110-X, (2001).

[Palensky 2000] Palensky, P. and Sauter, T., "Modular Software Architecture for Networked Appliances," In Proceedings of *(IWNA'02) 2nd International Workshop on Networked Appliances*, Rutgers University, IEEE Computer Society.

[Palet 2004a] Palet, J., "6Power: How to reach all the planets with IP," In Proceedings of *IEEE International Symposium on Applications and the Internet Workshop (SAINT)*, pp.120-126, Tokyo, Japan, IEEE Computer Society, (January 2004).

[Palet 2004b] Palet, J., Bano, L., Herandez, F. J., Marin, I., Manzano, D. M. and Moreno, J. J. P., "PLC-Based Home Automation System Completed," Spain, http://www.6power.org/open/6power_pu_d4_10_v1_4.pdf, (February 2004).

[Paolucci 2002a] Paolucci, M., Kawamura, T., Payne, T. R. and Sycara, K., "Semantic Matching of Web Services Capabilities," In Proceedings of *The First International Semantic Web Conference (ISWC)*, pp.333-347, Sardinia, Italy, Springer, (June 2002).

[Paolucci 2002b] Paolucci, M., Kawamura, T., Payne, T. R. and Sycara, K., "Importing the Semantic Web in UDDI," In Proceedings of *Web Services, E-Business, and the Semantic Web - CAiSE 2002 International Workshop (WES'02)*, pp.225-236, Toronto, Ontario, Canada, Springer-Verlag, (May 2002).

[Paolucci 2003] Paolucci, M., Sycara, K. and Kawamura, T., "Delivering Semantic Web Services," In Proceedings of *The Twelfth International World Wide Web Conference*, pp.111-118, Budapest, Hungary, (May 2003).

[Parameswaran 2001] Parameswaran, M., Susarla, A. and Whinston, A. B., "P2P Networking: An information-Sharing Alternative.," *IEEE Computer*, vol.34 (7), pp.31-38, (2001).

[Pattenden 2001] Pattenden, S., Colebrook, P., Ungar, S., Borghese, F., Francon, C. and Ambrosio, R., "Architecture for HomeGate, the residential gateway (AHRG)," http://hes-standards.org/doc/SC25_WG1_N0912.doc, (2001).

[Free Peers 2005] Free Peers, "BearShare," http://www.bearshare.com/, (Accessed: 2006).

[Poltavets 2005] Poltavets, Y., Part, Y. and Kim, D., "IEEE1394 to UPnP Software Bridge Structure," In Proceedings of *IEEE International Conference on Consumer Electronics (ICCE)*, pp.375-376, Las Vegas, Nevada, USA, IEEE Computer Society, (January 2005).

[Qu 2001] Qu, C. and Nejdl, W., "Exploring JXTASearch for P2P Learning Resource Discovery," Learning Lab Lower Saxony, Hannover, http://citeseer.ist.psu.edu/qu01exploring.html, (November 2001).

[Ratnasamy 2001] Ratnasamy, S., Fancis, P., Handley, M. and Karp, R., "A Scalable Content-Addressable Network," In Proceedings of *ACM SIGCOMM annual conference of the Special Interest Group on Data Communications*, pp.161-172, San Diego, California, USA, ACM Press, (August 2001).

[Roberts 1967] Roberts, L. G., "Multiple Computer Networks and Intercomputer Communication," In Proceedings of *Proceedings of the ACM symposium on Operating System Principles*, pp.3.1-3.6, ACM Press, (January 1967).

[Rose 2001] Rose, B., "Home Networks: A Standards Perspective," *IEEE Communications Magazine*, vol.39 (12), pp.78-85, (December 2001).

[Rowstron 2001] Rowstron, A. and Druschel, P., "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," In Proceedings of *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp.329-350, Heidelberg, Germany, ACM Press, (November 2001).

[Rumsey 2003] Rumsey, D., "Statistics for Dummies," Wiley Publishing, Inc., Hoboken, NJ, 07030, ISBN: 0764554239.

[Shareaza 2005] Shareaza Development Team, "Shareaza," Web Site, Shareaza Open Source Development Team, http://www.shareaza.com/, (Accessed: 2006).

[Shigeoka 2002] Shigeoka, I., "Instant Messaging in Java - The Jabber Protocols," Manning Publications Co., 209 Bruce Park Avenue, Greenwich, CT 06830, ISBN: 1-930110-46-4.

[Siebes 2002] Siebes, R. and van Harmelen, F., "Ranking Agent Statements for Building Evolving Ontologies," In Proceedings of *Workshop on Meaning Negotiation, in conjunction with the Eighteenth National Conference on Artificial Intelligence*, Emonton, Alberta, Canada, (July 2002).

[Sirin 2003] Sirin, E., Hendler, J. A. and Parasia, B., "Semi-automatic Composition of Web Services using Semantic Descriptions," In Proceedings of *(WSMAI'03) Proceedings of the 1st Workshop on Web Services: Modelling, Architecture and Infrastructure*, pp.17-24, Angers, France, ICEIS Press, (April 2003).

[Siuru 2000] Siuru, B., "Appliances on the Internet," *Poptronics*, vol.1, pp.41-44, (June 2000).

[Smith 2000] Smith, E., Dominguez, M. and Merabti, M., "Component Support Services for Heterogeneous Networked Appliances," In Proceedings of *(IWNA'02) 2nd IEEE International Workshop on Networked Appliances*, Rutgers University, IEEE Computer Society.

[Smith 2005] Smith, M. K., Welty, C. and McGuinness, D. L., "OWL Web Ontology Language," http://www.w3.org/TR/owl-guide/, (Accessed: 2006).

[Stephens 2001] Stephens, L. M. and Huhns, M. N., "Consensus Ontologies - Reconciling the Semantics of Web Pages and Agents," *IEEE Internet Computing*, vol.5 (5), pp.92-95, (Sep/Oct 2001).

[Stephens 2003] Stephens, L. M., Gangam, A. K. and Huhns, M. N., "Constructing Consensus Ontologies for the Semantic Web: A Conceptual Approach," Klewers Academic Publishers, (2003).

[Sycara 2003] Sycara, K., Paolucci, M., Ankolekar, A. and Srinivasan, N., "Automated discovery, interaction and composition of Semantic Web services," *Web Semantics*, vol.1 (1), pp.27-46, (December 2003).

[JINI Technology 2005] JINI Technology, "JINI Technology," Web Page, http://www.jini.org/, (Accessed: 2006).

[France Telecom 2005] France Telecom, "ePerSpace: Towards the era of personal services at home and everywhere," http://www.ist-eperspace.org/, (Accessed: 2006).

[Traversat 2002] Traversat, B., Abdelaziz, M., Duigou, M., Hugly, J., Pouyoul, E. and Yeager, B., "Project JXTA Virtual Network," 901 San Antonio Road, Palo Alto, CA 94303, USA, http://www.jxta.org/project/www/docs/JXTAprotocols_01nov02.pdf, (October 2002).

[Traversat 2003] Traversat, B., Abdelaziz, M. and Pouyoul, E., "Project JXTA: A Loosely-Consistent DHT Rendezvous Walker," http://www.jxta.org/docs/jxta-dht.pdf, (2003).

[Travert 2004] Travert, S. and Lemonier, M., "The MediaNet Project," In Proceedings of *5th International Workshop on Image Analysis for Multimedia Interactive Services*, Lisboa, Portugal, (April 2004).

[Tsarkov 2005] Tsarkov, D. and Horrocks, I., "FaCT++," http://owl.man.ac.uk/factplusplus/, (Accessed: 2006).

[Ungar 2000] Ungar, S. G., "The VHN Network," In Proceedings of *IEEE Second International Workshop on Networked Appliances, IWNA'2000*, Rutgers University, (2000).

[Stanford University 2005a] Stanford University, "Protege OWL Plugin API," http://protege.stanford.edu/plugins/owl/api/index.html, (Accessed: 2006).

[Stanford University 2005b] Stanford University, "Using the Protege-OWL Reasoning API," http://protege.stanford.edu/plugins/owl/api/ReasonerAPIExamples.html, (Accessed: 2006).

[Uschold 1996] Uschold, M. and Gruninger, M., "Ontologies: Principles, Methods and Applications," *The Knowledge Engineering Review*, vol.11 (2), pp.93-155, (June 1996).

[VESA 2005] VESA, "Video Electronics Standards Association," http://www.vesa.org, (Accessed: 2005).

[W3C 2004] W3C, "OWL Web Ontology Language Guide," Web Site, W3C, http://www.w3.org/TR/owl-guide/, (Accessed: 2006).

[W3C 2005] W3C, "World Wide Web Consortium," http://www.w3c.org, (Accessed: 2006).

[Waterhouse 2001] Waterhouse, S., "JXTA Search: Distributed Search for Distributed Networks," http://search.jxta.org/JXTAsearch.pdf, (Accessed: 2006).

[WebMethods 2003] WebMethods, "GLUE," http://www.webmethods.com/meta/default/folder/0000005452, (Accessed: 2006).

[Williams 2001] Williams, L. V., "CEA-851 versatile home network (VHN)-a home intranet backbone for inter-cluster connectivity using IEEE 1394 and IP," In Proceedings of *International Conference on Consumer Electronics, ICCE2001*, pp.230 - 231, (2001).

[Wilson 2002] Wilson, B. J., "JXTA," New Riders Publishing, 201 West 103rd Street, Indianapolis, Indiana 46290, ISBN: 0734712344.

[Zahariadis 2002] Zahariadis, T. and Pramataris, K., "Multimedia home networks: standards and interfaces," *Computer Standards and Interfaces*, vol.24 (5), pp.425-235, (2002).

[Zahariadis 2003] Zahariadis, T. B., "Home Networking Technologies and Standards," Artech House, Inc., Boston - London, ISBN: 1580536484, (2003).

APPENDIX A: NASUF USE CASE DIAGRAMS

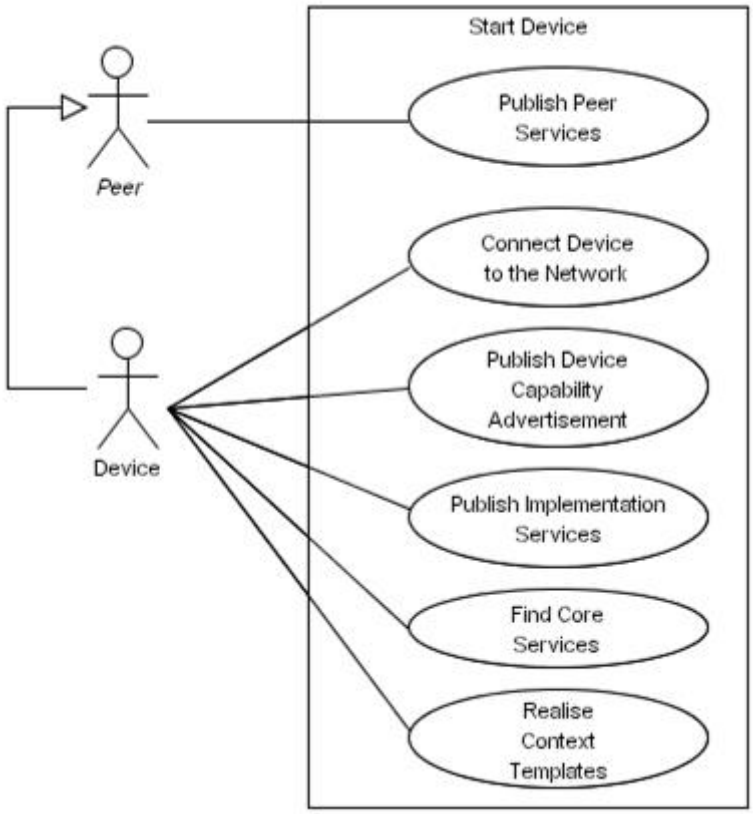

Figure A.1 Start Device

Description:

This Use Case illustrates a typical scenario when a Device is initially switched on within this framework.

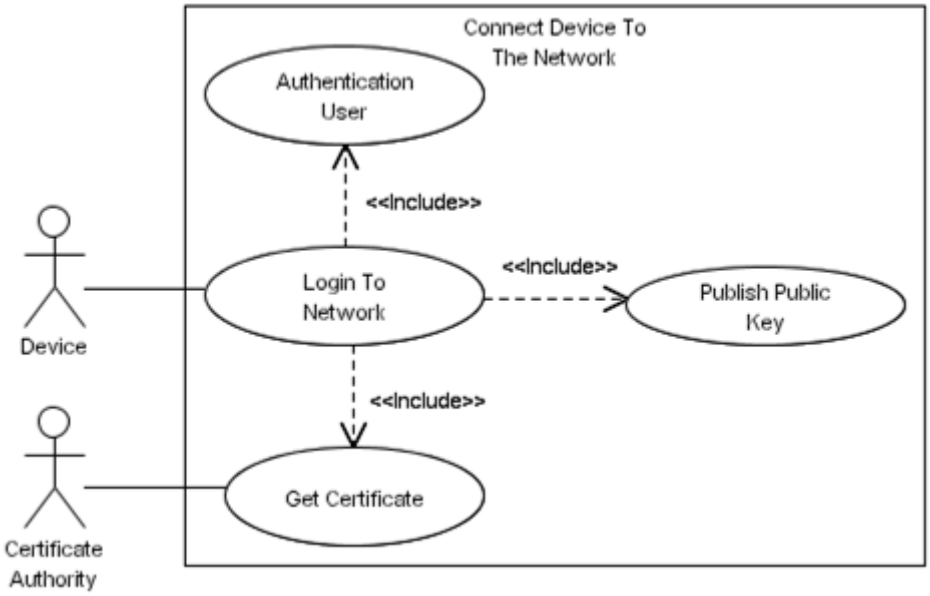

Figure A.2 Connect Device to Network

Description:

This Use Case illustrates how a device is connected to the network within this framework.

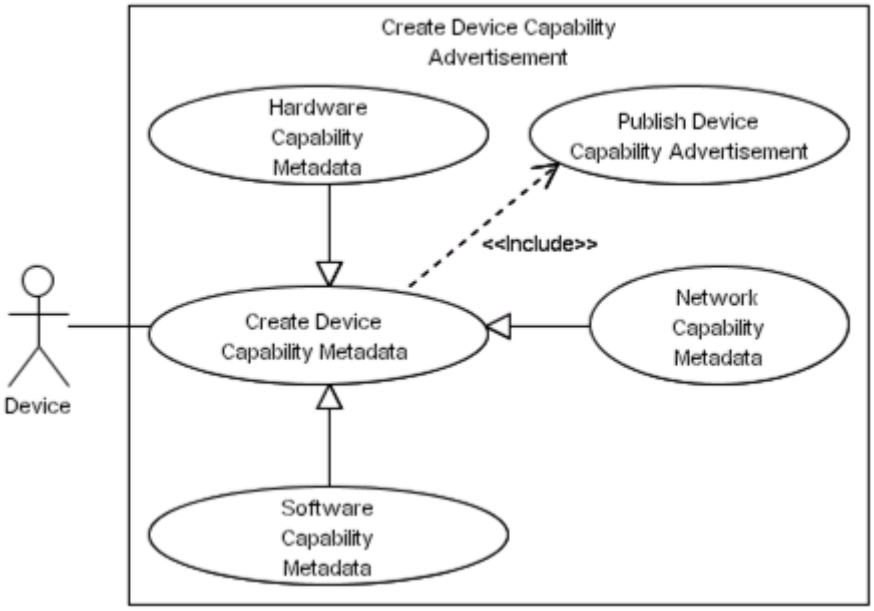

Figure A.3 Create Device Capability Model

Description:

This Use Case illustrates how the capabilities of the device are described in terms of the devices hardware, software and network capabilities within this framework.

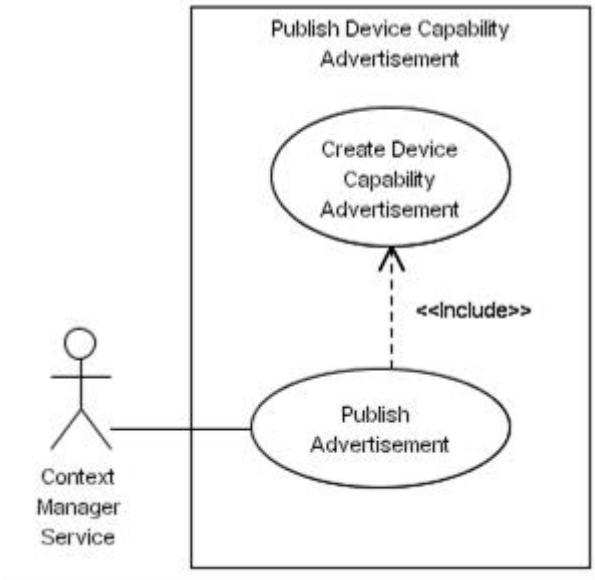

Figure A.4 Publish Create Device Capability Model

Description:

This Use Case illustrates how a device publishes a capability advertisement locally and globally within this framework.

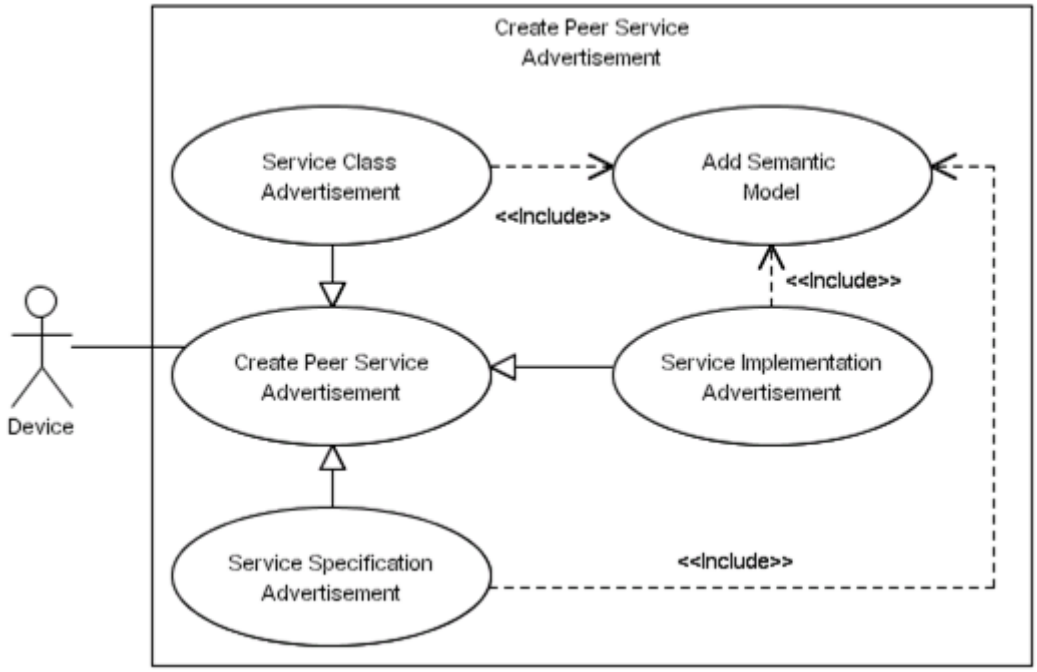

Figure A.5 Create Peer Service Advertisement

Description:

This Use Case illustrates how Peer Service Advertisements are created within this framework.

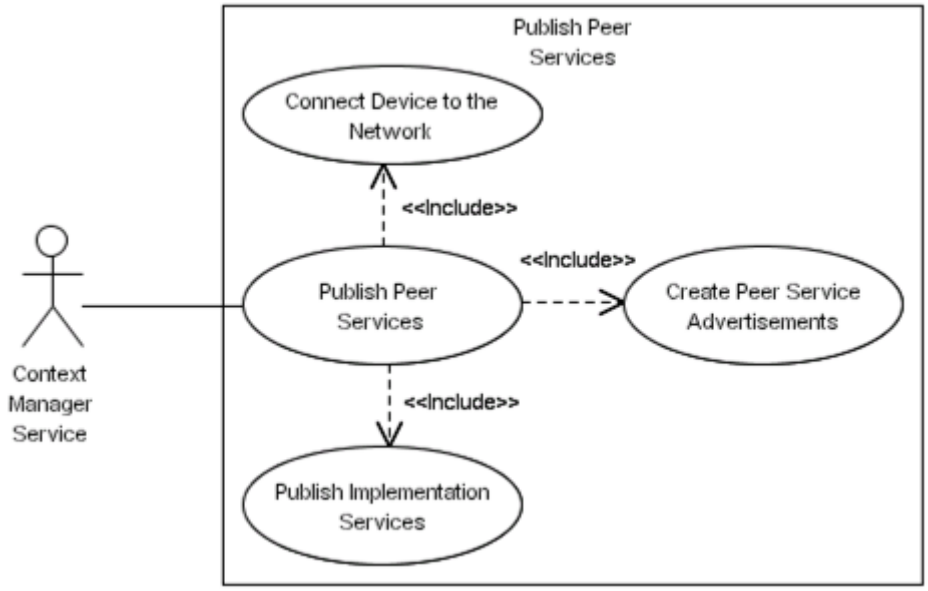

Figure A.6 Publish Peer Service

Description:

This Use Case illustrates how peer services are published locally and remotely within the P2P network in this framework.

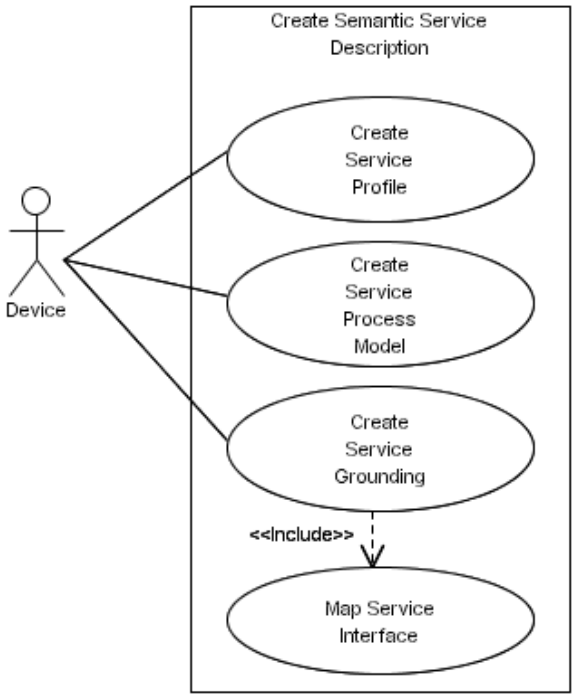

Figure A.7 Create Semantic Service Models

Description:

This Use Case illustrates how semantic service descriptions are created by devices, which are explicitly mapped to service interfaces.

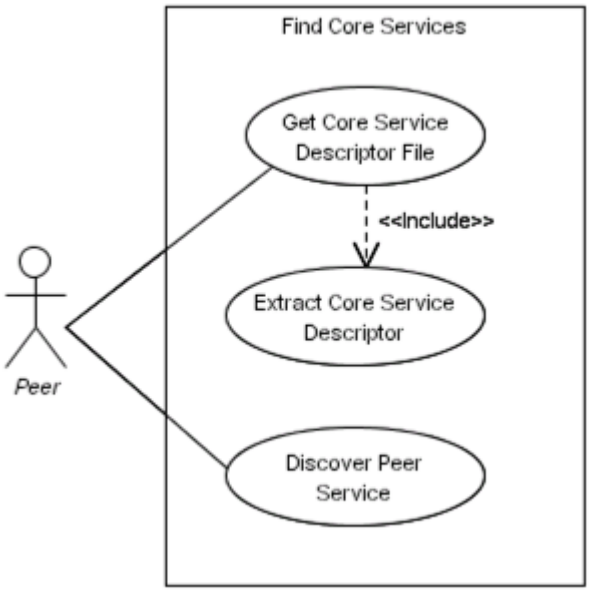

Figure A.8 Find Core Services

Description:

This Use Case illustrates how a device finds core service advertisements located on the device and distributed within the P2P network.

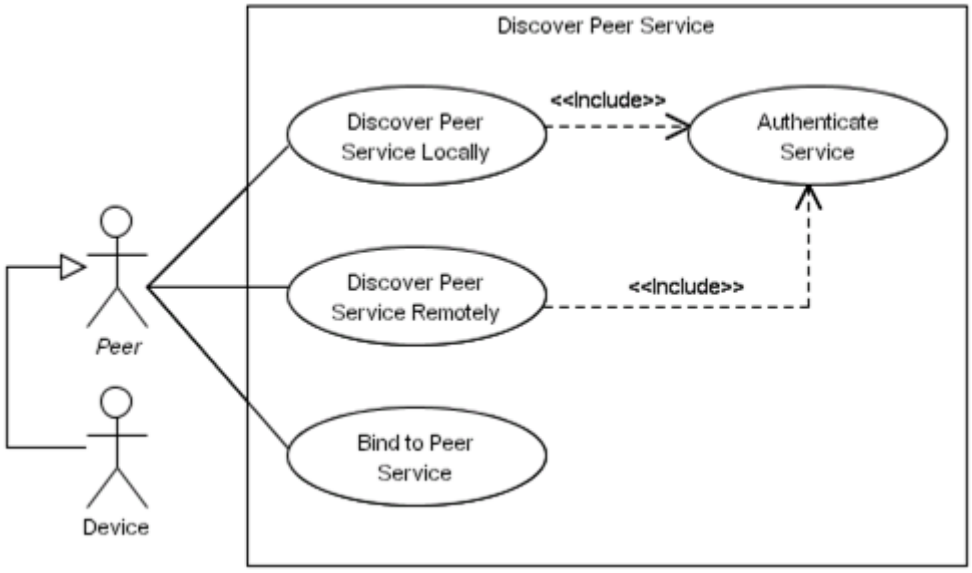

Figure A.9 Discover Peer Service

Description:

This Use Case illustrates how peer services are discovered within this framework.

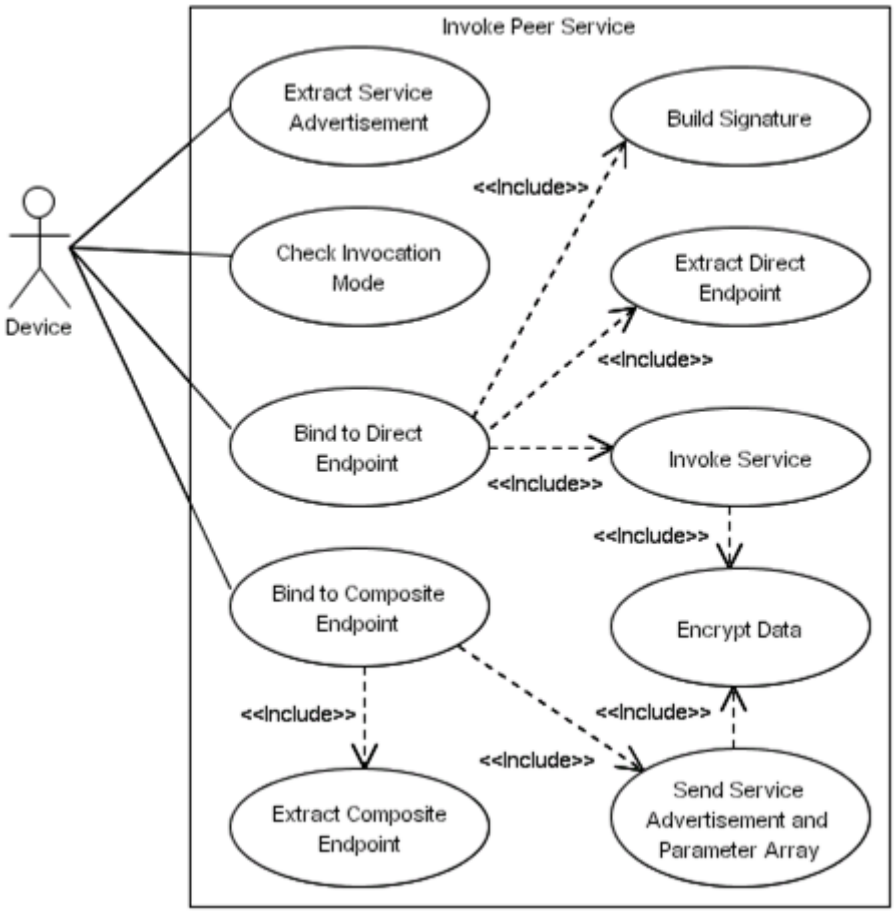

Figure A.10 Invoke Peer Service

Description:

This Use Case illustrates how peer services are invoked within this framework.

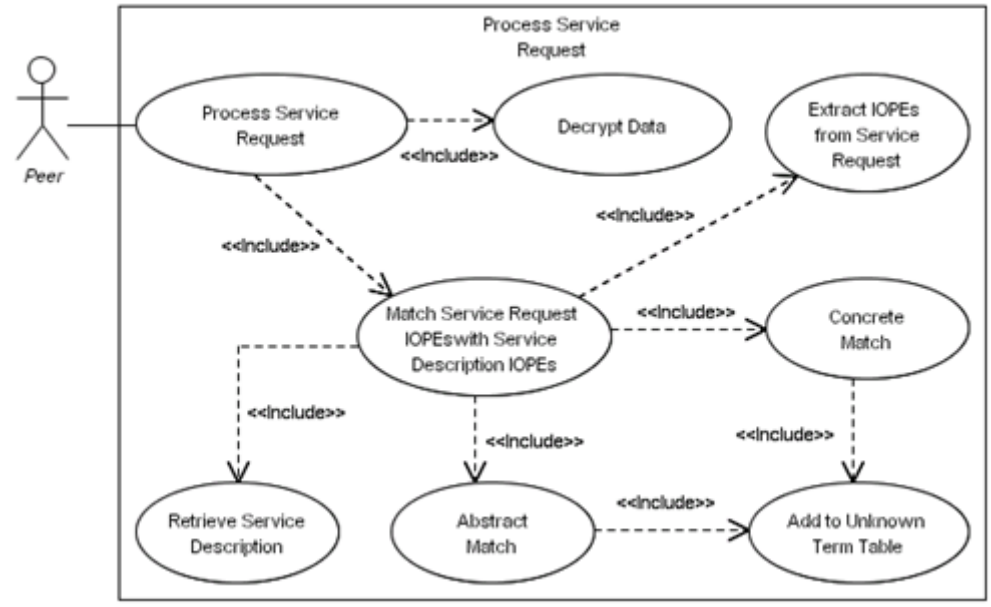

Figure A.11 Process Service Request

Description:

This Use Case illustrates how a device processes a service request received from the P2P network within this framework.

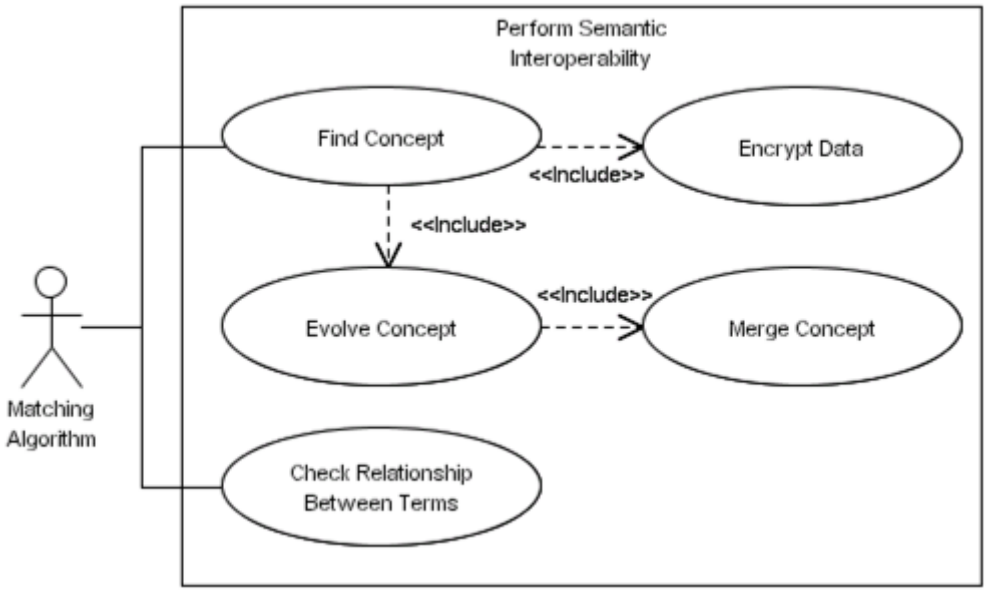

Figure A.12 Perform Semantic Interoperability

Description:

This Use Case illustrates how the matching algorithms perform semantic interoperability between terms that are syntactically different but semantically equivalent.

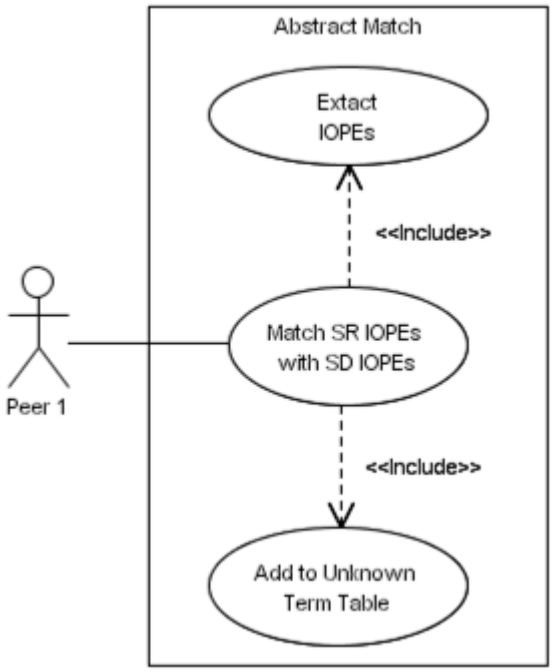

Figure A.13 Perform Abstract Match

Description:

This Use Case illustrates how the abstract matching algorithm matches the Inputs, Outputs, Preconditions, and Effects (IOPEs) found within both the service request and the service description within this framework.

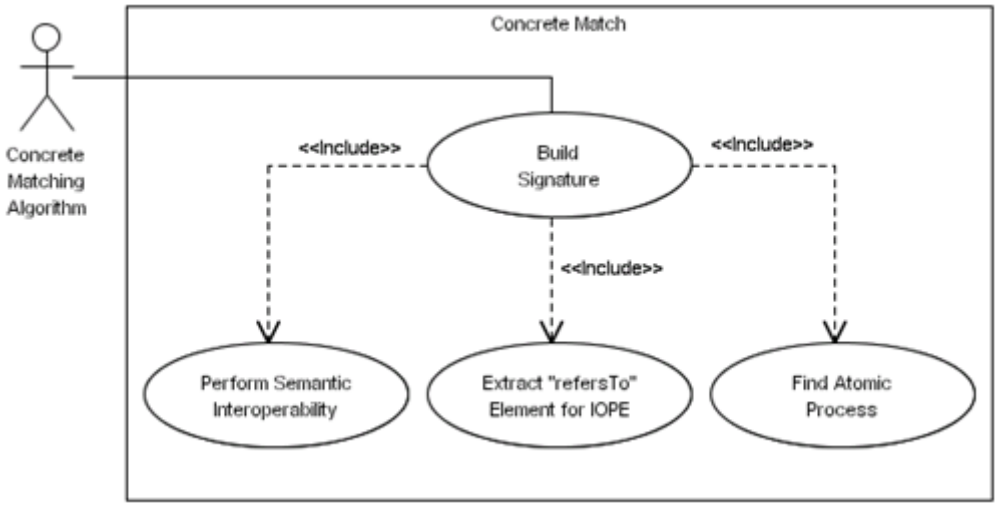

Figure A.14 Perform Concrete Match

Description:

This Use Case illustrates how the concrete matching algorithm matches the Inputs and Outputs found in the service request with concrete bindings found in the service interface within this framework.

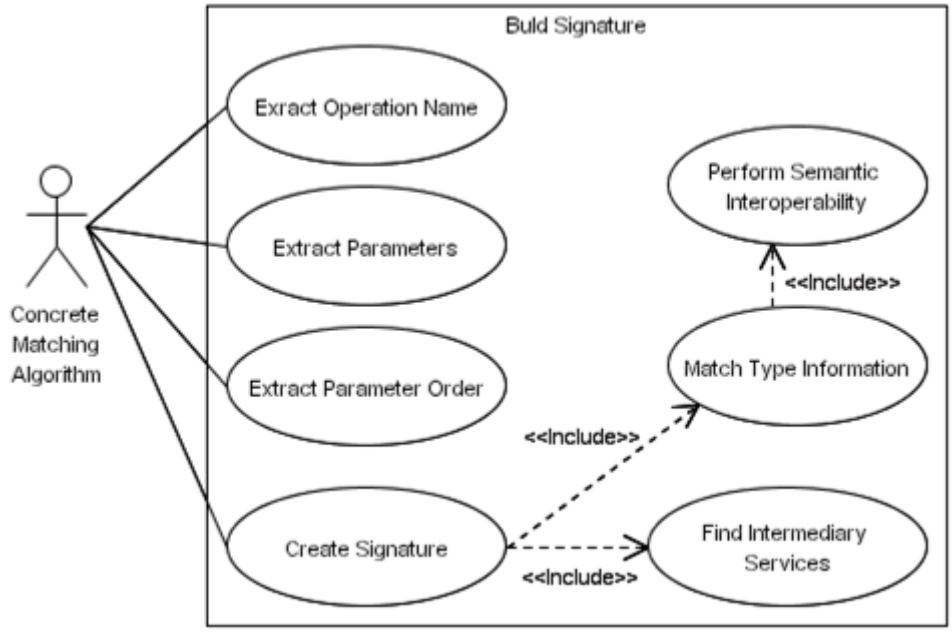

Figure A.15 Build Signature

Description:

This Use Case illustrates how signatures are built to determine if a concrete match has been found within this framework.

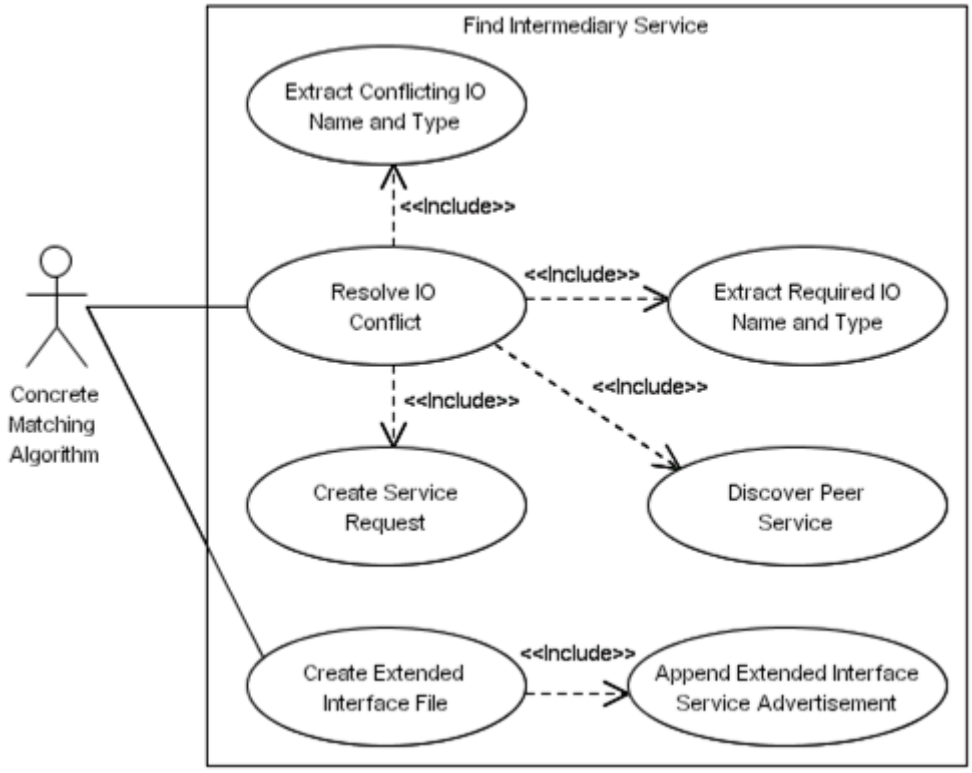

Figure A.16 Find Intermediary Service

Description:

This Use Case illustrates how intermediary services are found and how extended interface files are created within this framework.

APPENDIX B: NASUF CLASS DIAGRAMS

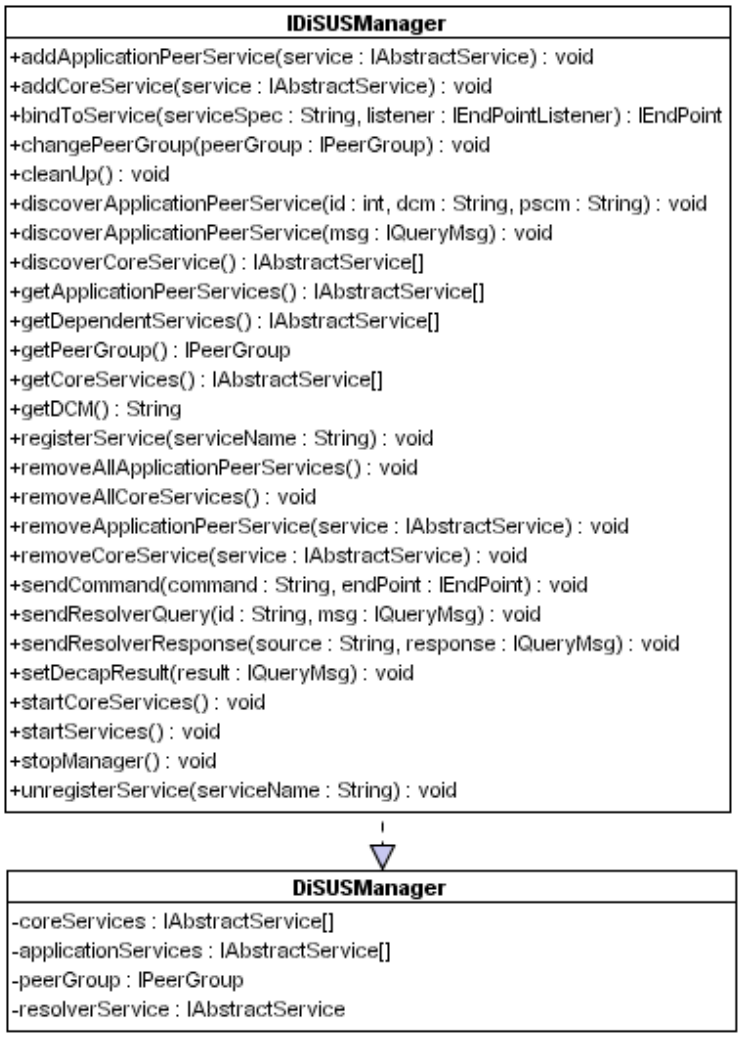

Figure B.1 Distributed Semantic Unstructured Services Manager

Description:

This Class Diagram illustrates the classes used to implement Distributed Semantic Unstructured Services within this framework.

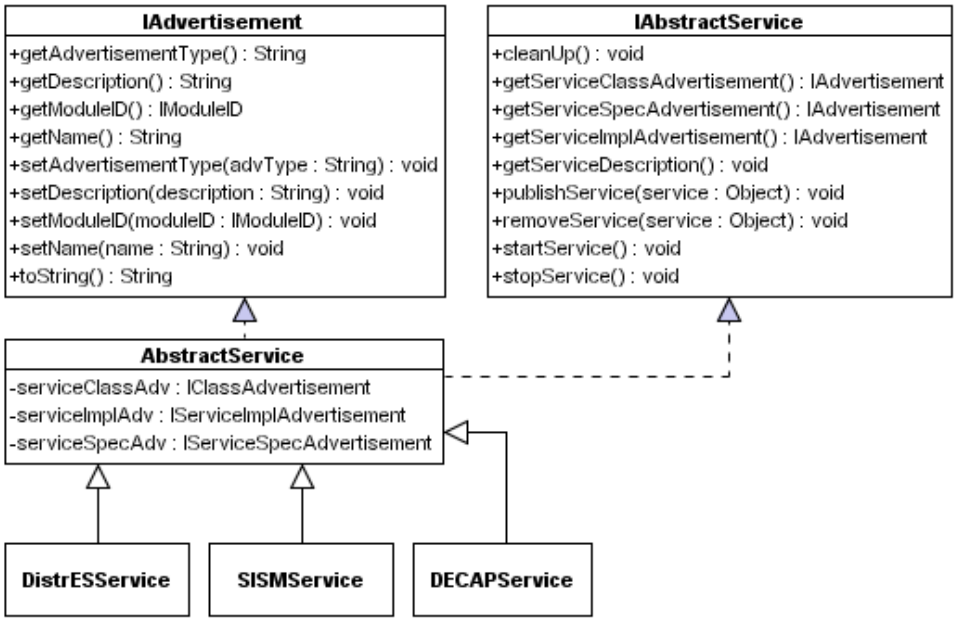

Figure B.2 Peer Service

Description:

This Class Diagram illustrates the required Peer Service class, including its associated subclasses which must be used within this framework.

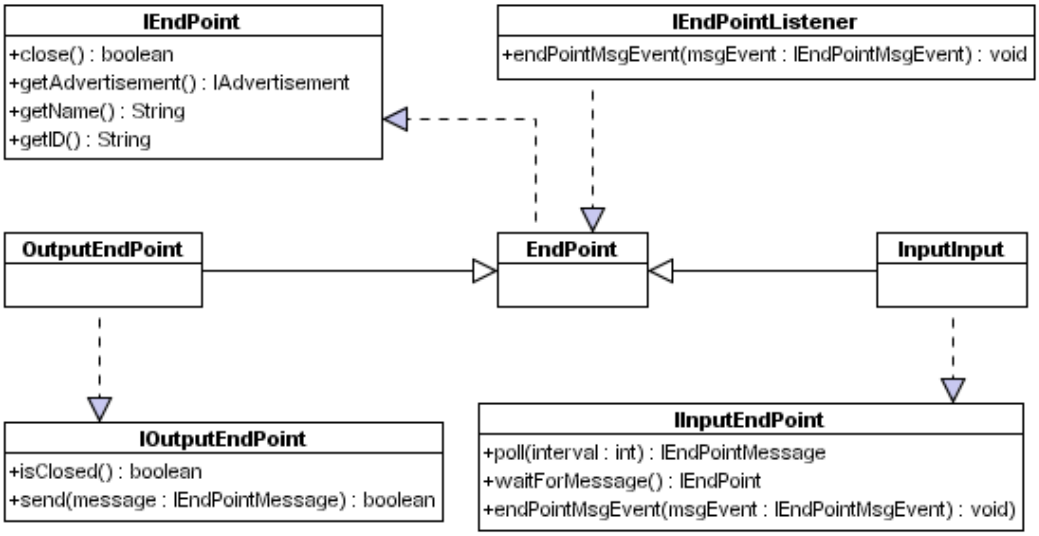

Figure B.3 Endpoint

Description:

This Class Diagram illustrates the classes required for the Endpoints used within this framework.

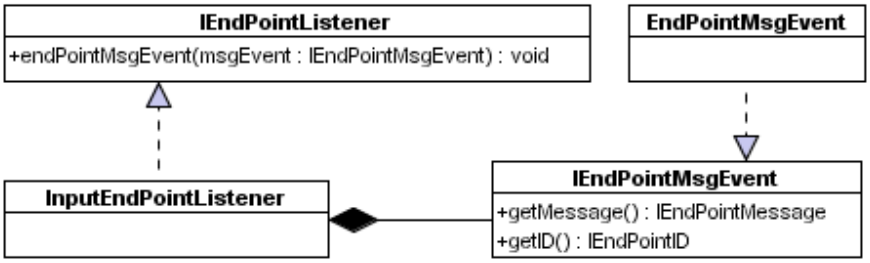

Figure B.4 Endpoint Listener

Description:

This Class Diagram illustrates the classes required for implementing input endpoint message listeners within this framework.

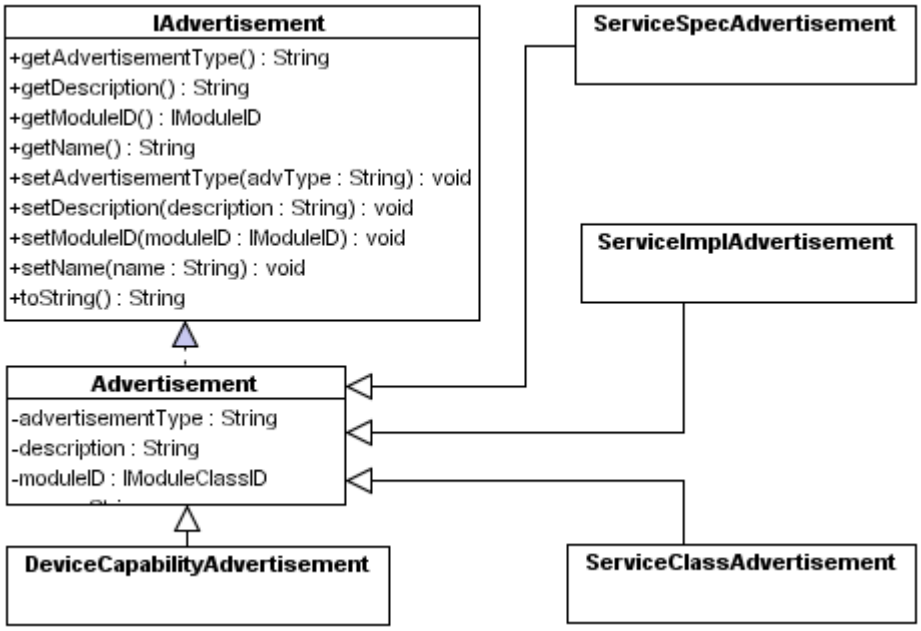

Figure B.5 Service Advertisement

Description:

This Class Diagram illustrates the required classes to create Service Advertisements within this framework.

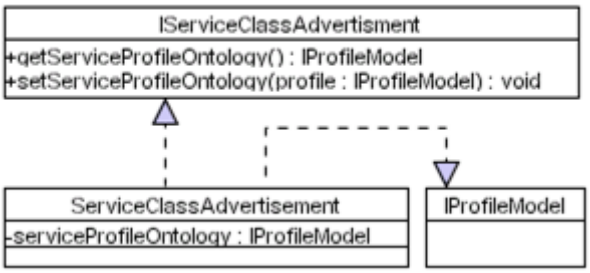

Figure B.6 Service Class Advertisement

Description:

This Class Diagram illustrates the classes required to create a Service Class Advertisement within this framework.

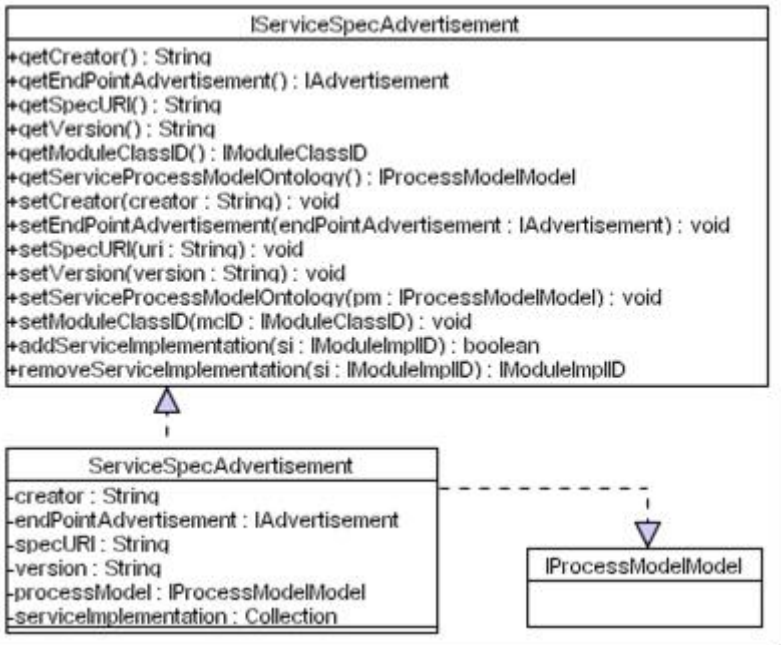

Figure B.7 Service Specification Advertisement

Description:

This Class Diagram illustrates the required classes to create a service specification advertisement within this framework.

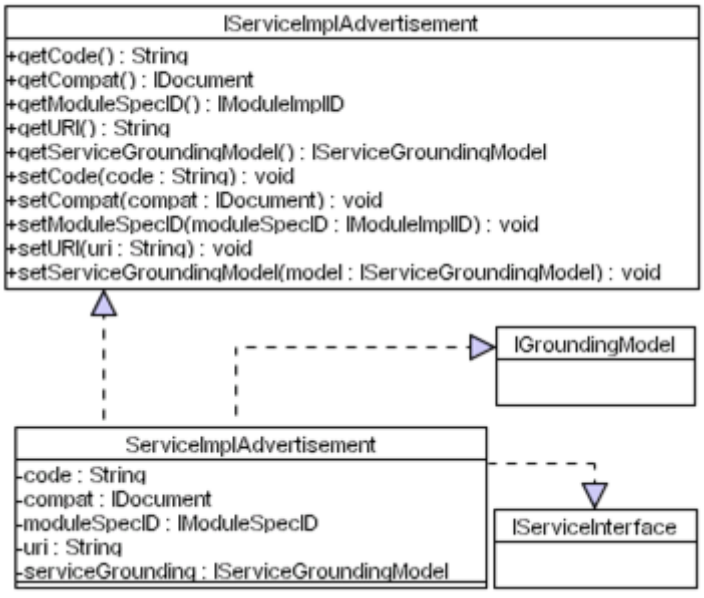

Figure B.8 Service Implementation Advertisement

Description:

This Class Diagram illustrates the required classes to create a service implementation service within this framework.

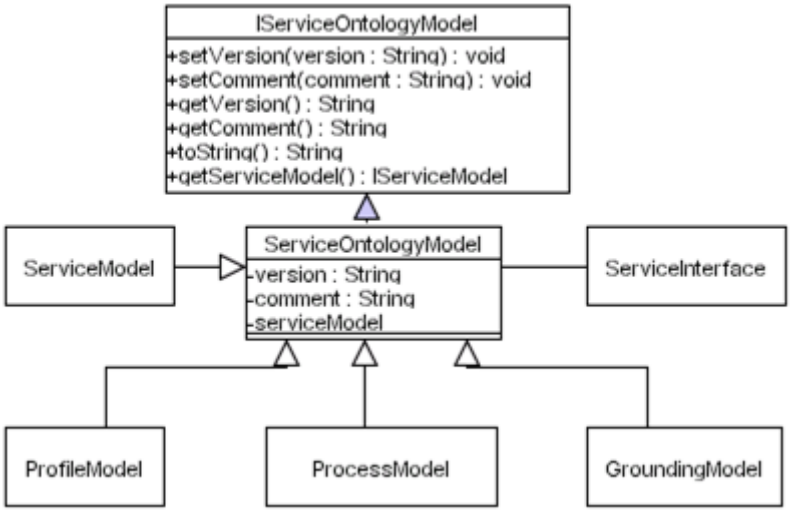

Figure B.9 Service Ontology Model

Description:

This Class Diagram illustrates the required classes to create a Service Ontology Model within this framework.

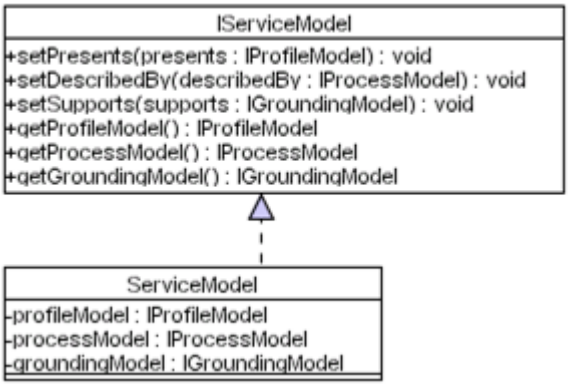

Figure B.10 Service Model

Description:

This Class Diagram illustrates the required classes needed to create a Service Model within this framework.

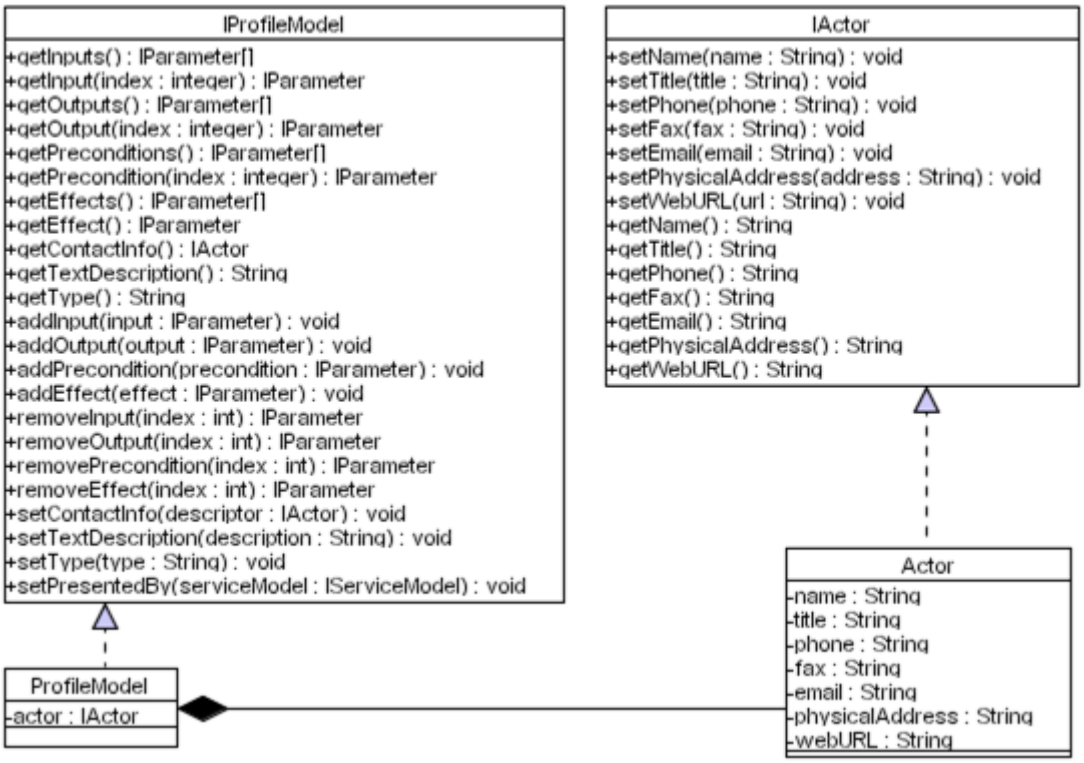

Figure B.11 Service Profile Model

Description:

This Class Diagram illustrates the required classes need to create a Service Profile Model within this framework.

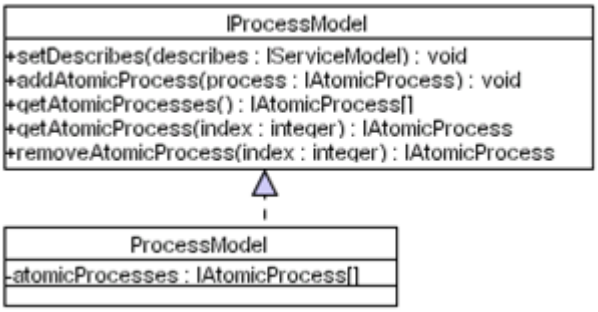

Figure B.12 Service Process Model

Description:

This Class Diagram illustrates the classes required to create a service process ontology model within this framework.

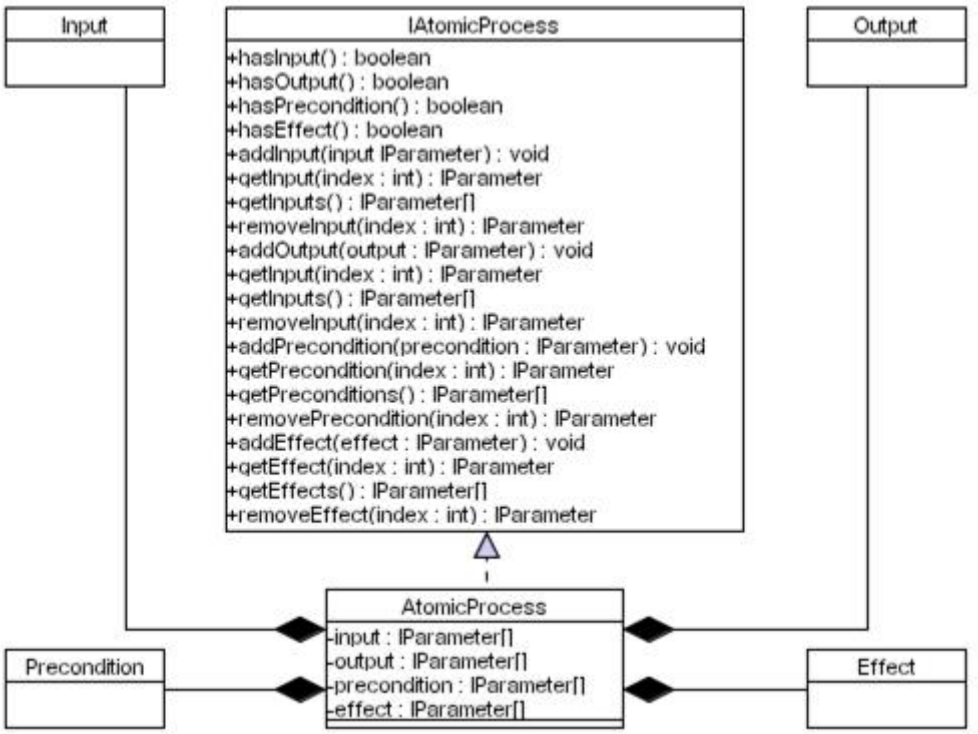

Figure B.13 Atomic Process

Description:

This Class Diagram illustrates the classes required to create an Atomic Process within this framework.

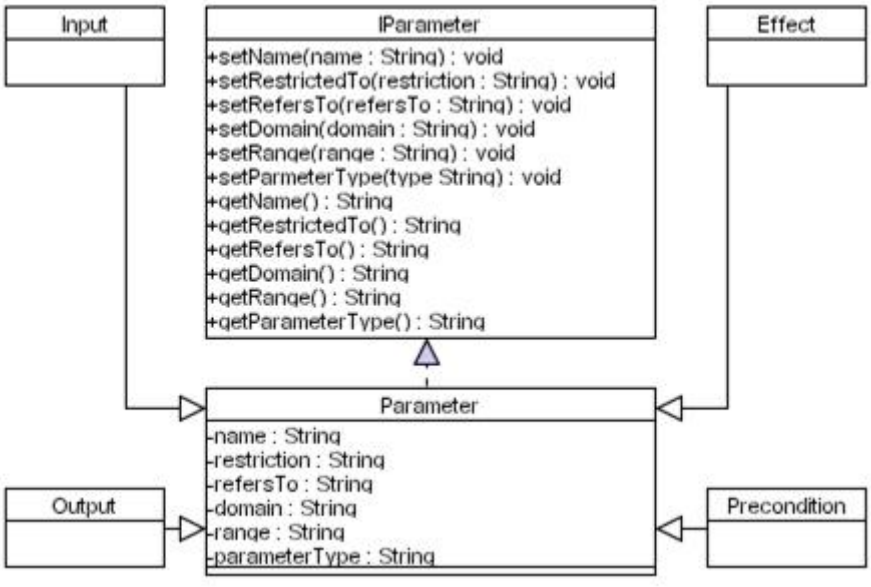

Figure B.14 Parameter

Description:

This Class Diagram illustrates the classes required to create a Parameter within this framework.

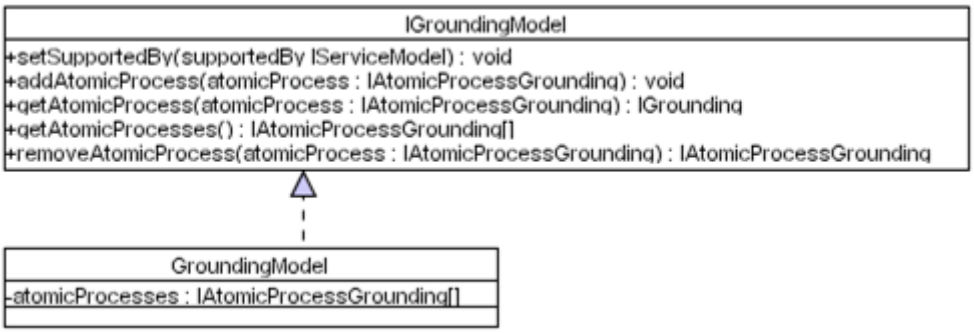

Figure B.15 Service Grounding Model

Description:

This Class Diagram illustrates the classes required to create a service process ontology model within this framework.

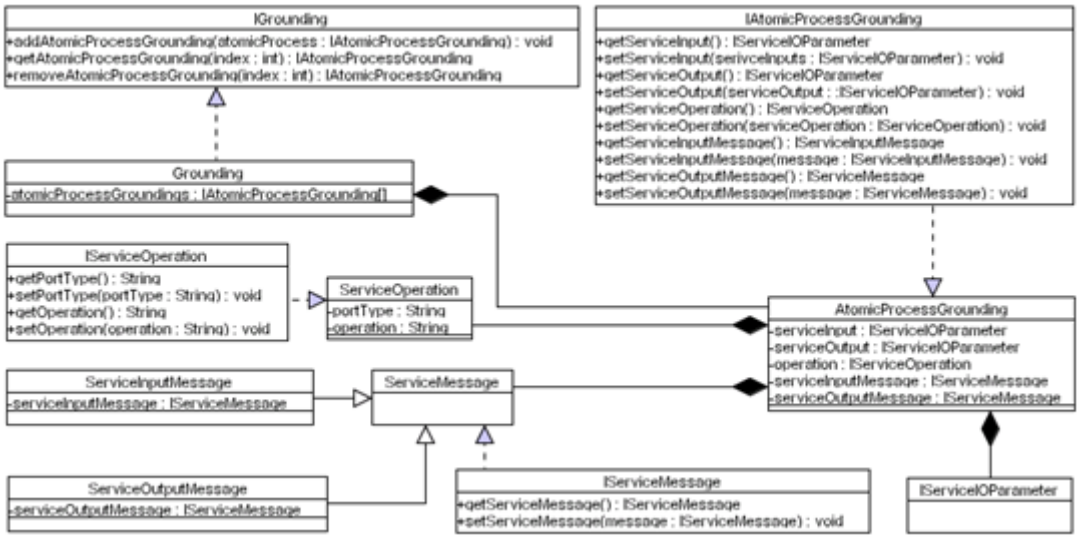

Figure B.16 Atomic Process Grounding

Description:

This Class Diagram illustrates the classes required to create an Atomic Process Grounding within this framework.

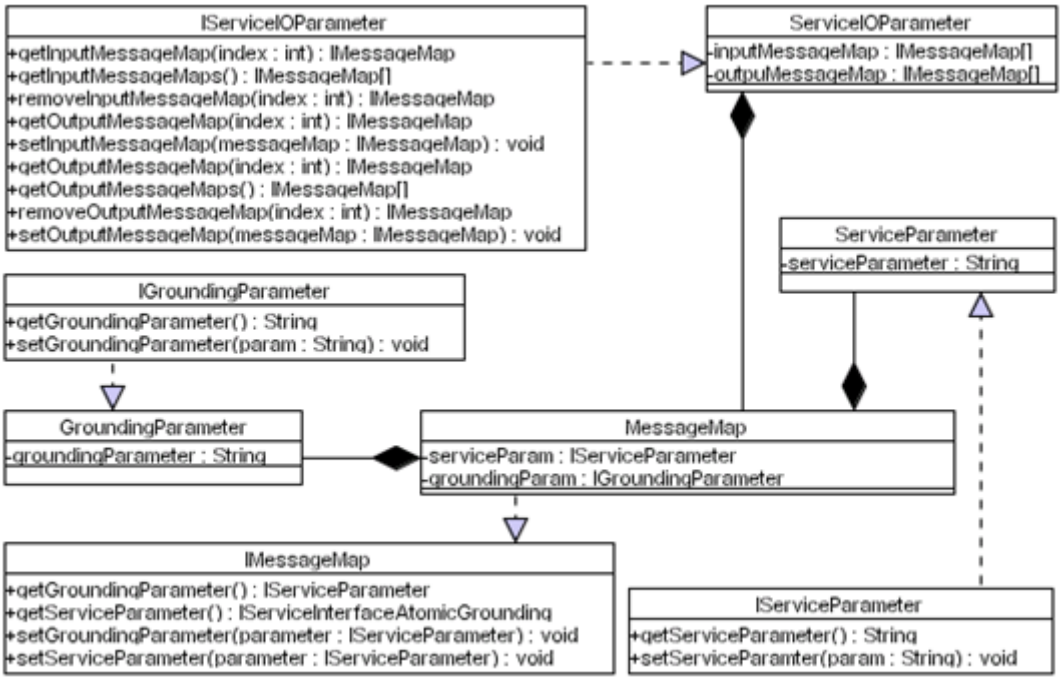

Figure B.17 Service Input/Output Parameter

Description:

This Class Diagram illustrates the classes required to create a Service Input/Output Parameter within this framework.

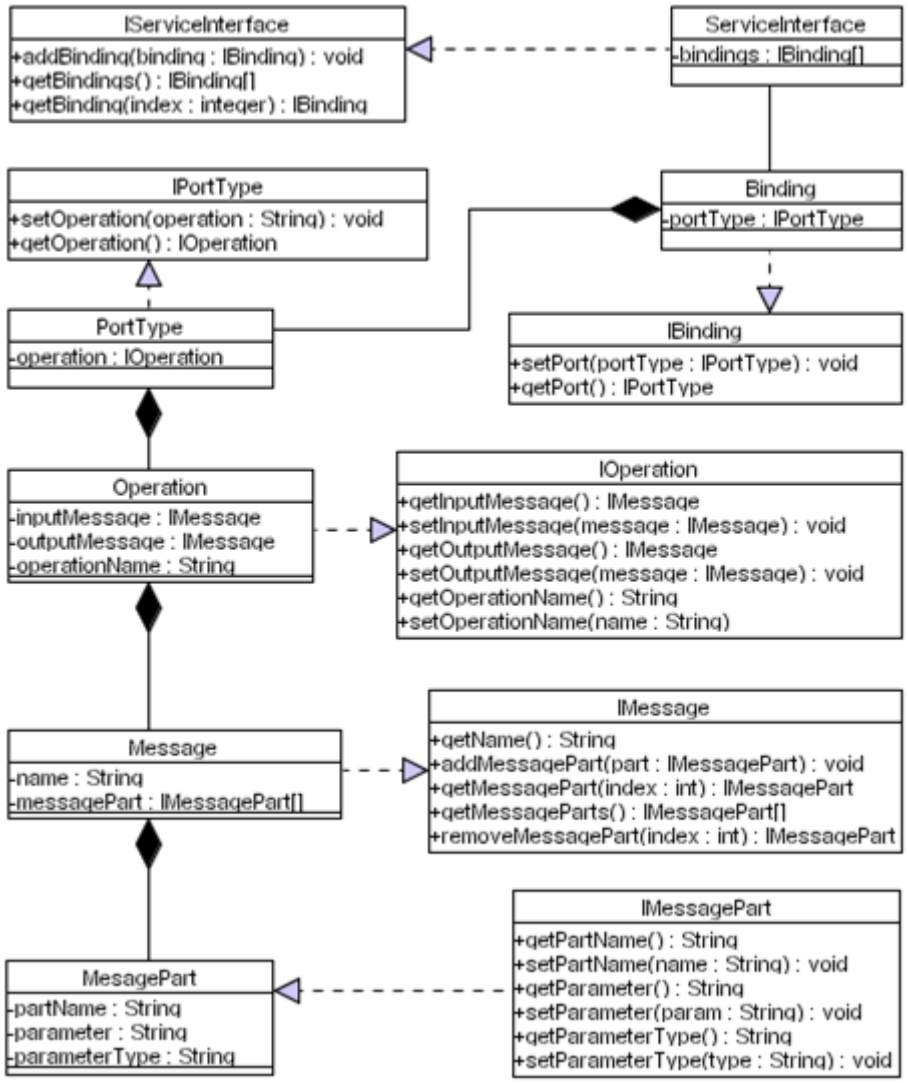

Figure B.18 Service Interface Model

Description:

This Class Diagram illustrates the classes required to create a service interface model within this framework.

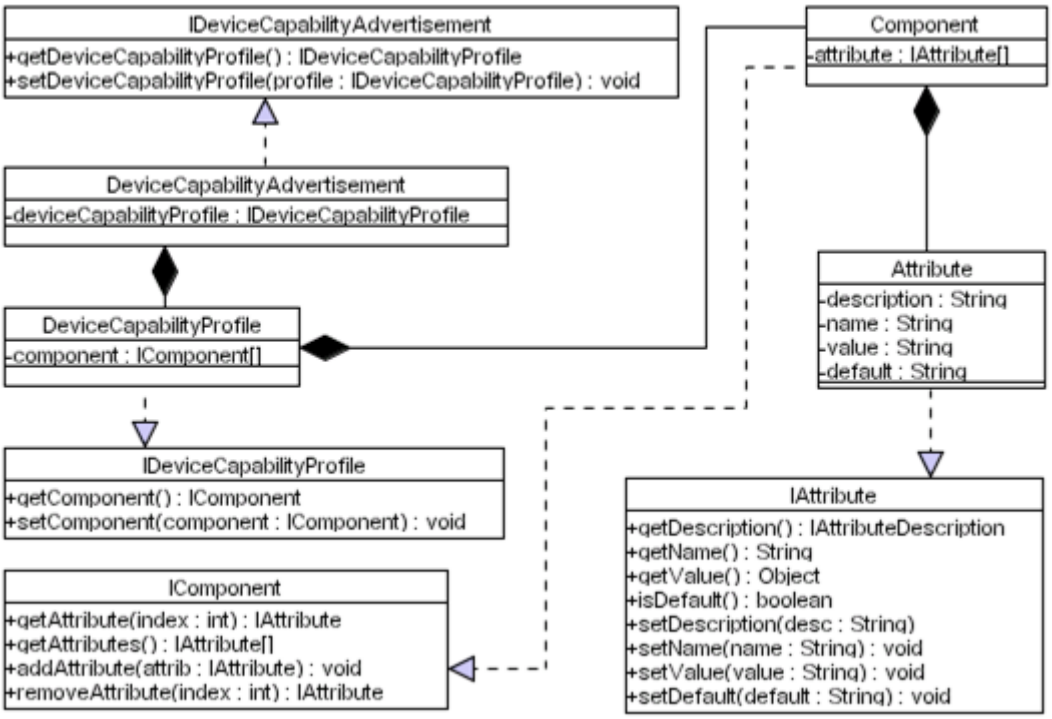

Figure B.19 Device Capability Model

Description:

This Class Diagram illustrates the classes required to create a device capability model within this framework.

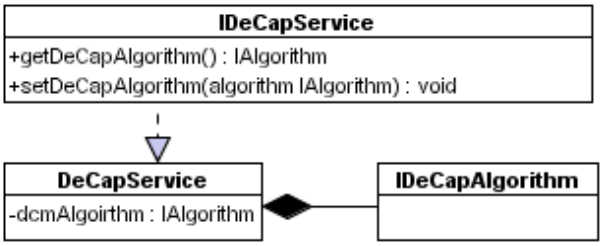

Figure B.20 Device Capability Service

Description:

This Class Diagram illustrates the classes required to create the DeCap Service within this framework.

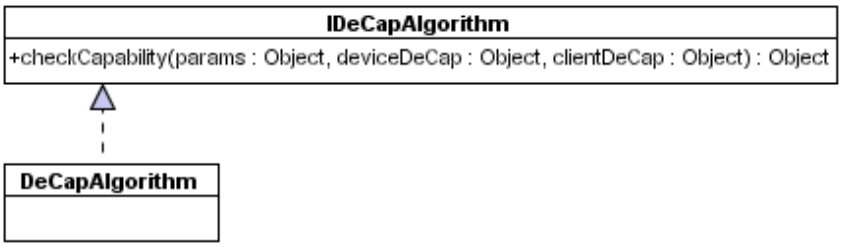

Figure B.21 Device Capability Algorithm

Description:

This Class Diagram illustrates the classes required to create the DeCap Service within this framework.

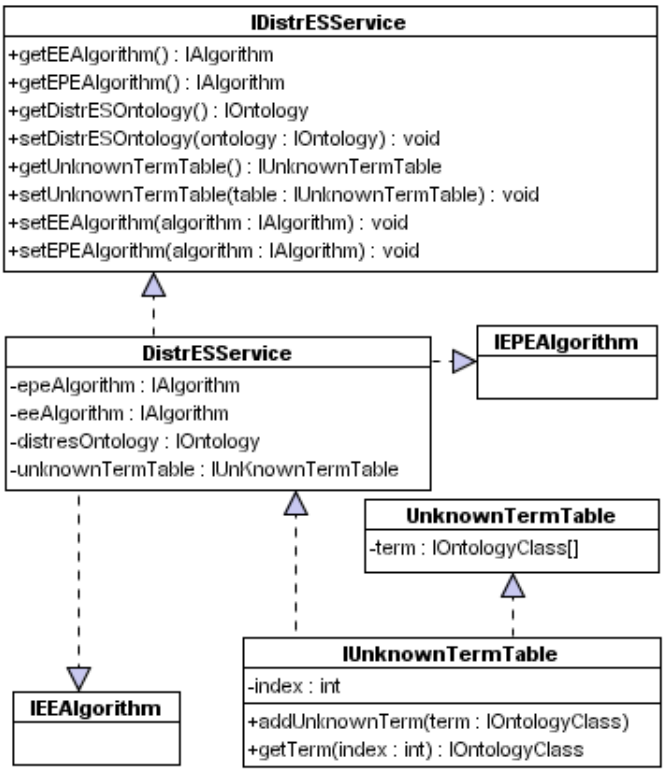

Figure B.22 Distributed Emergent Semantics Service

Description:

This Class Diagram illustrates the classes required for implementing distributed emergent semantics within this framework.

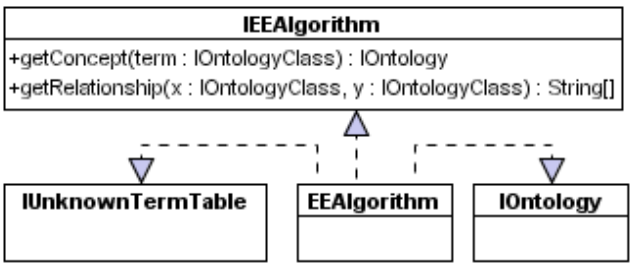

Figure B.23 Extraction Engine

Description:

This Class Diagram illustrates the classes required for the extracting concepts from the knowledge base within this framework.

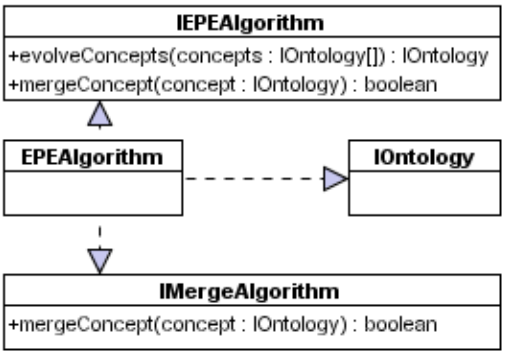

Figure B.24 Evolutionary Pattern Extraction Engine

Description:

This Class Diagram illustrates the classes required to extract common patterns and evolve knowledge structures within this framework.

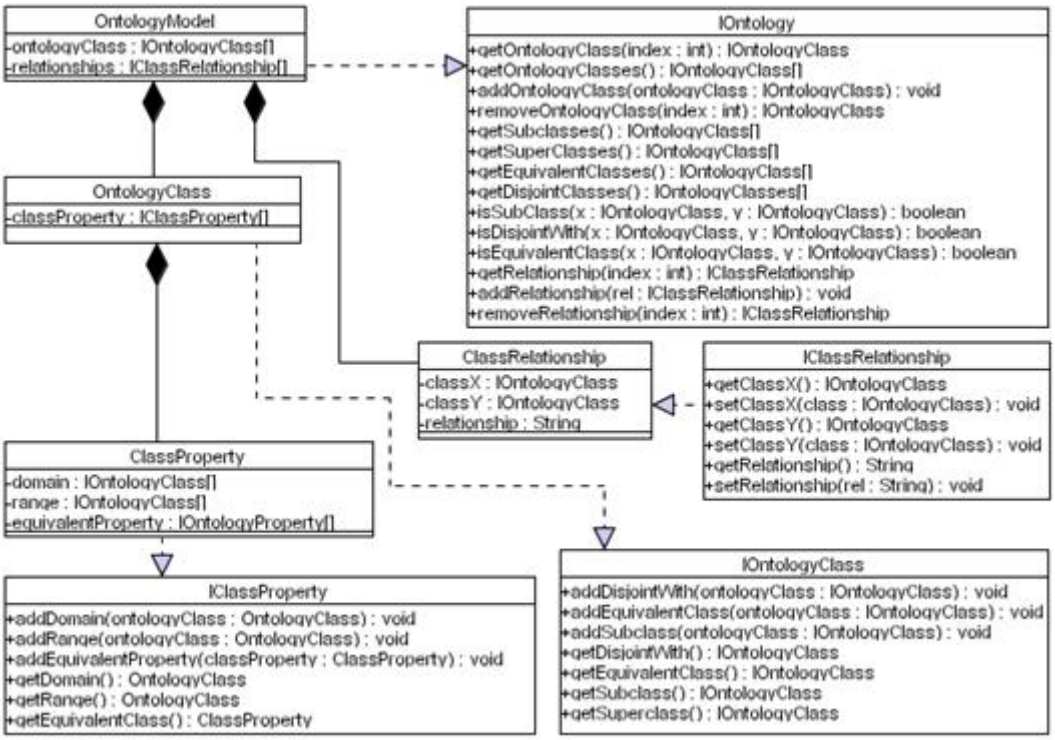

Figure B.25 DistrES Ontology

Description:

This Class Diagram illustrates the classes required to describe ontologies within this framework.

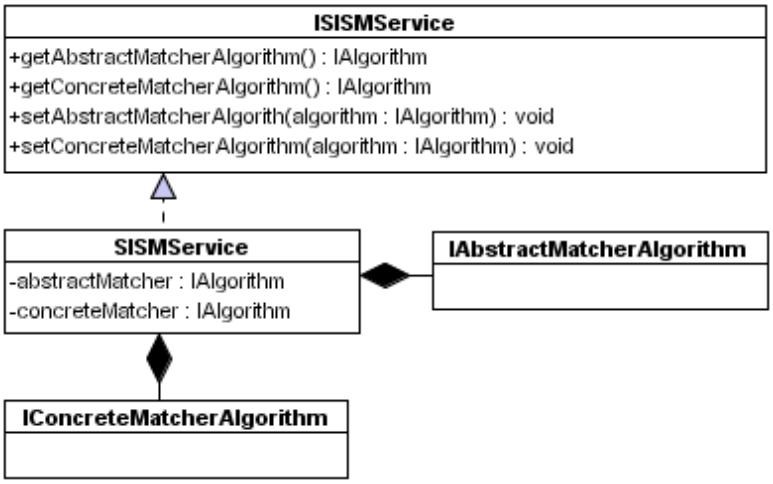

Figure B.26 SISM Service

Description:

This Class Diagram illustrates the required classes for performing semantic interoperability.

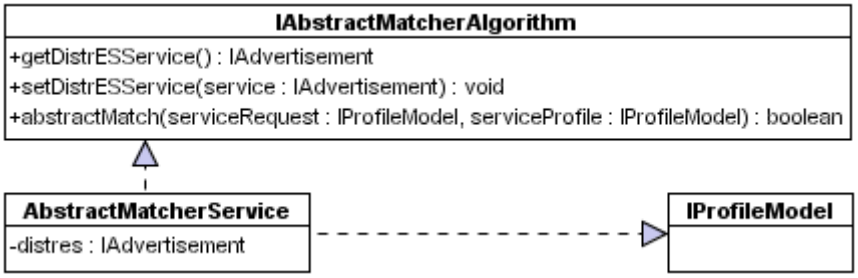

Figure B.27 Abstract Matcher Algorithm

Description:

This Class Diagram illustrates the classes required to abstract match service requests with service descriptions.

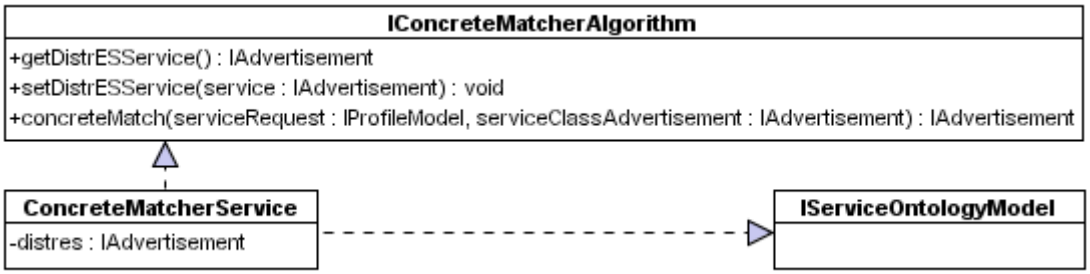

Figure B.28 Concrete Matcher Algorithm

Description:

This Class Diagram illustrates the required classes to concrete match service descriptions with signatures in service interfaces.

APPENDIX C: NASUF ACTIVITY DIAGRAMS

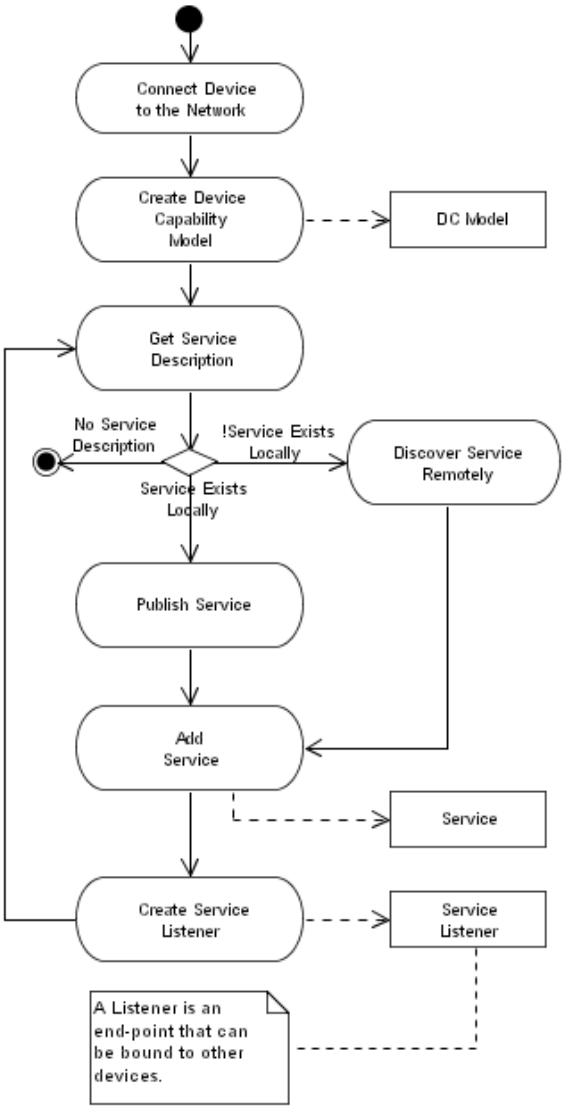

Figure C.1 Start Device

Description:

This Activity Diagram illustrates what happens when the device is initially started within this framework.

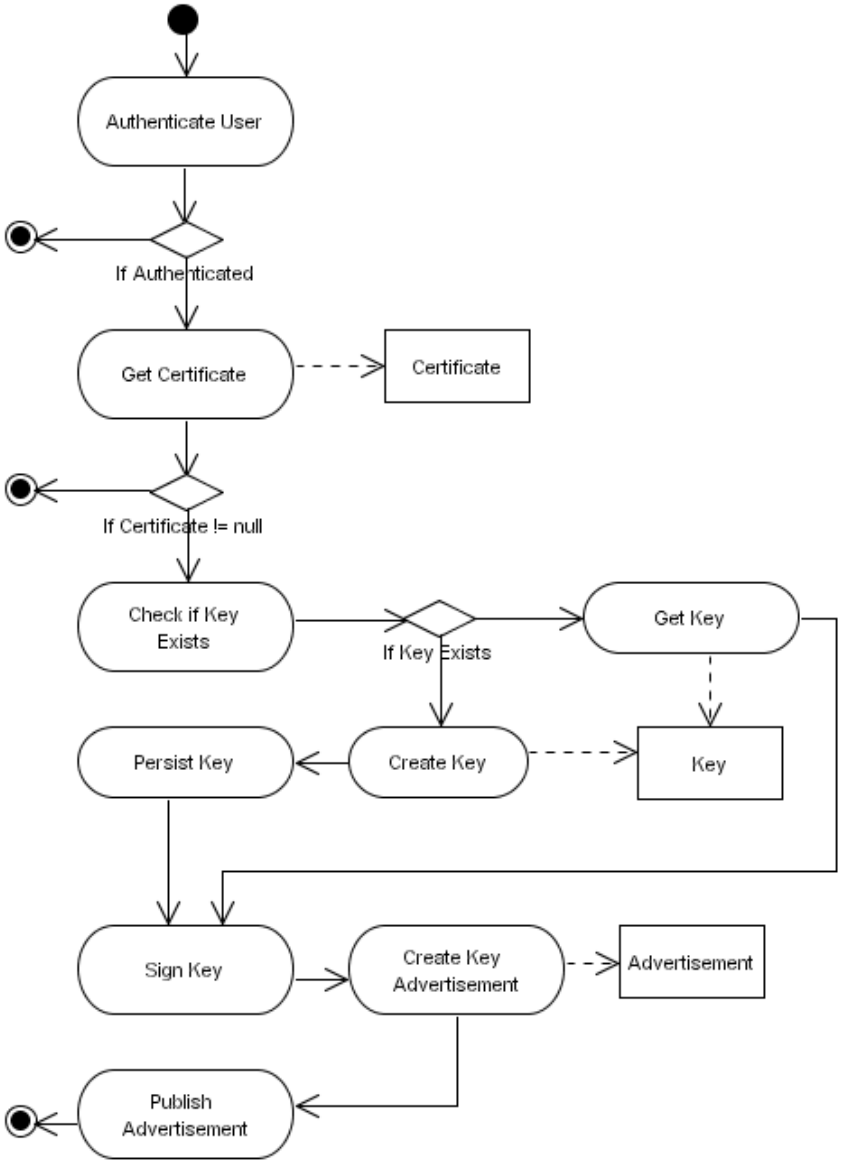

Figure C.2 Connect device to the network

Description:

This Activity Diagram illustrates how devices connect to the network within this framework.

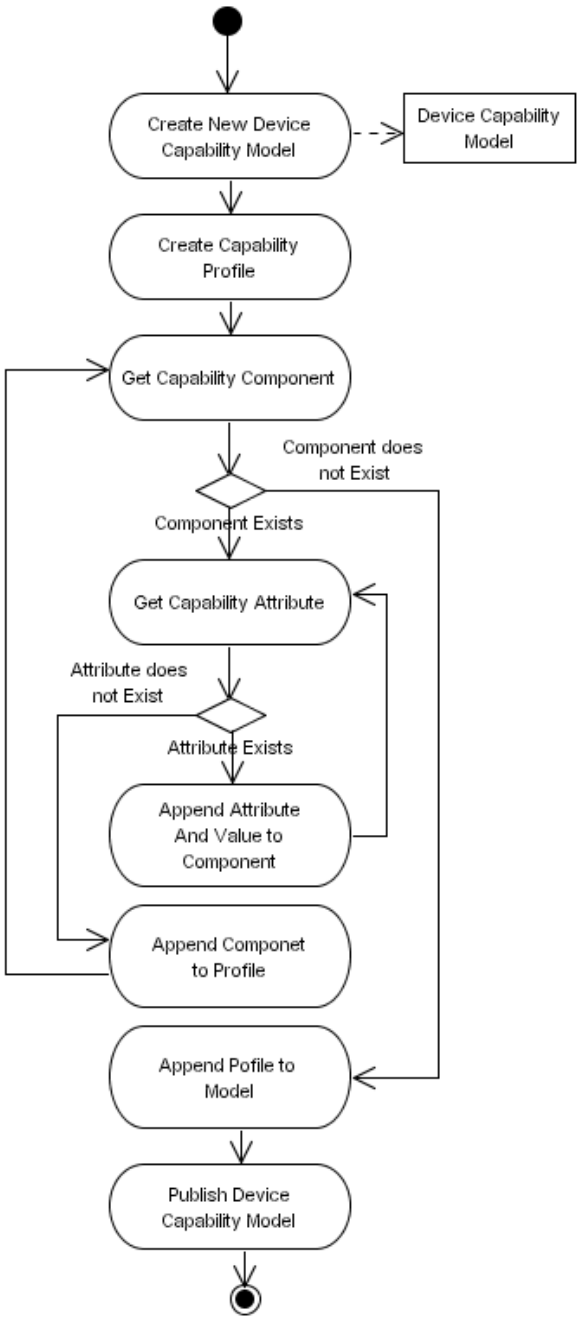

Figure C.3 Create device capability model

Description:

This Activity Diagram illustrates how a Device Capability Model is created within this framework.

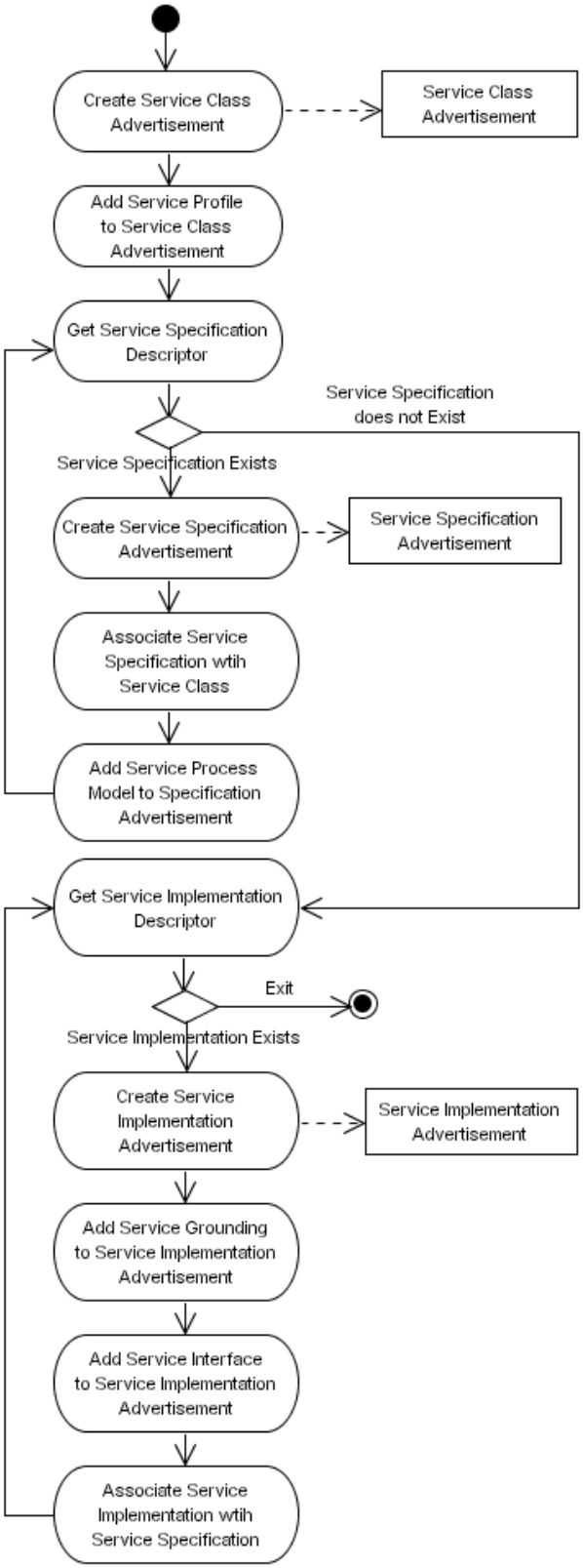

Figure C.4 Create Peer Service advertisements

Description:

This Activity Diagram illustrates how Peer Service Advertisements are created within this framework.

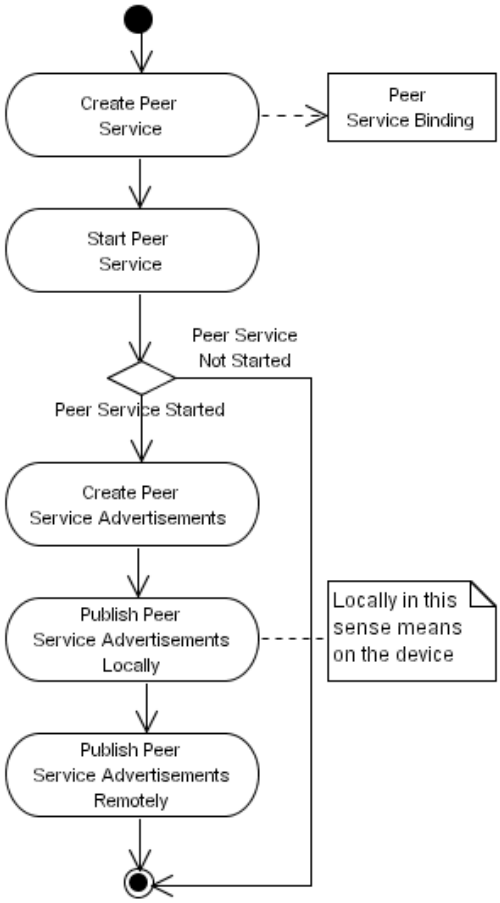

Figure C.5 Publish Peer Services

Description:

This Activity Diagram illustrates how Peer Services are published within this framework.

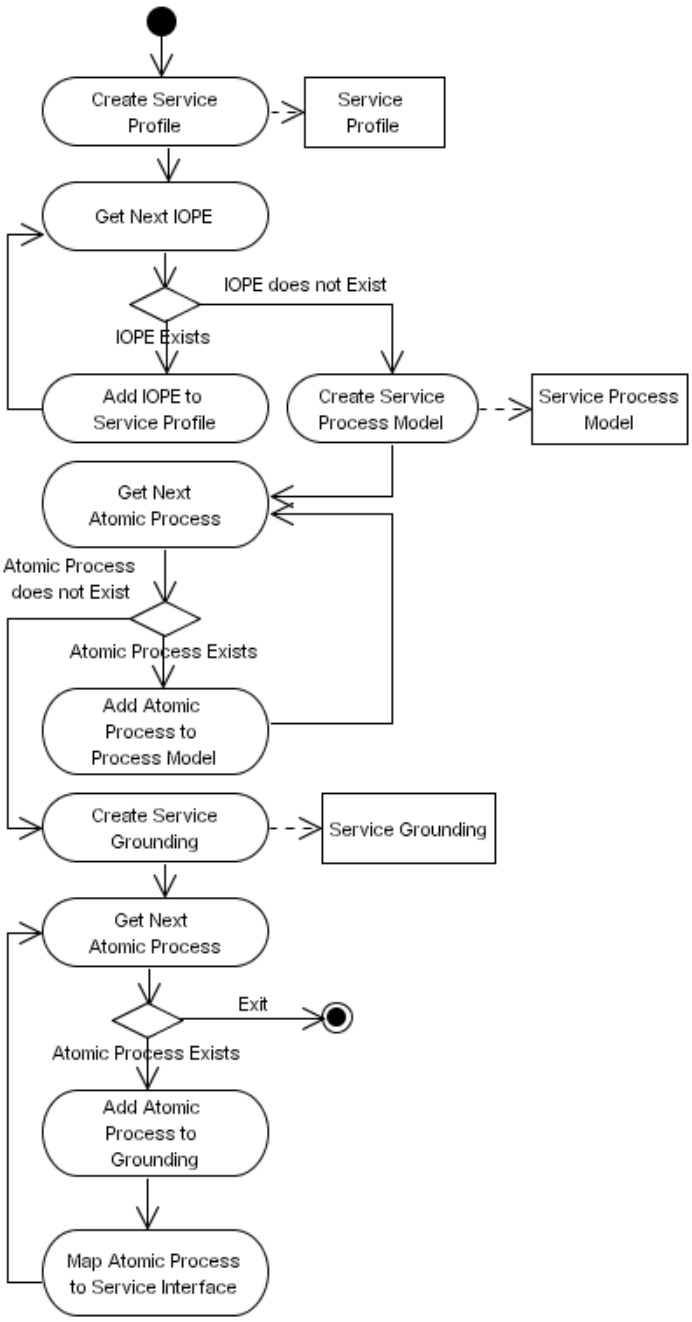

Figure C.6 Create Semantic Models

Description:

This Activity Diagram illustrates how semantic models are created within this framework.

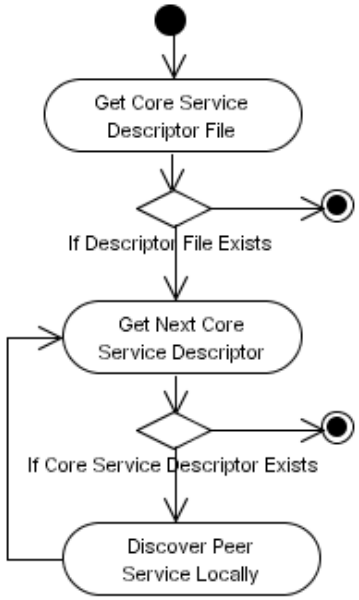

Figure C.7 Find Core Services

Description:

This Activity Diagram illustrates how core services are discovered within this framework.

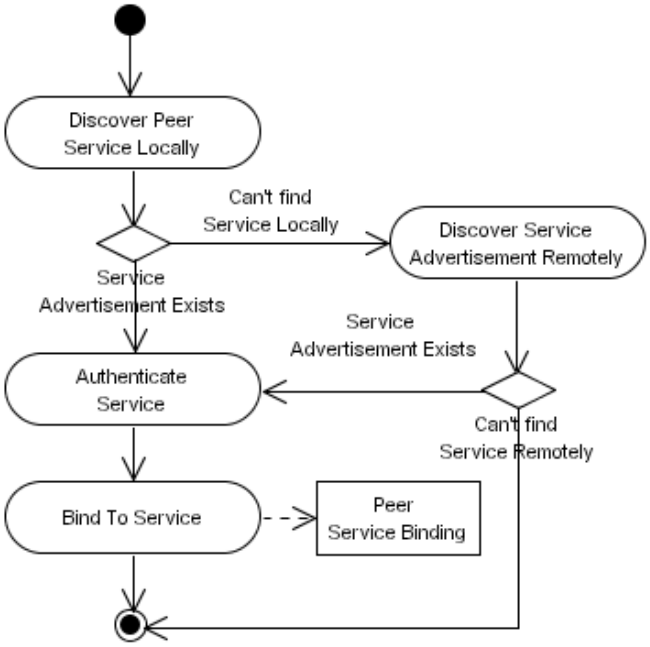

Figure C.8 Discover Peer Service

Description:

This Activity Diagram illustrates how Peer Services are discovered locally and remotely in the P2P network within this framework.

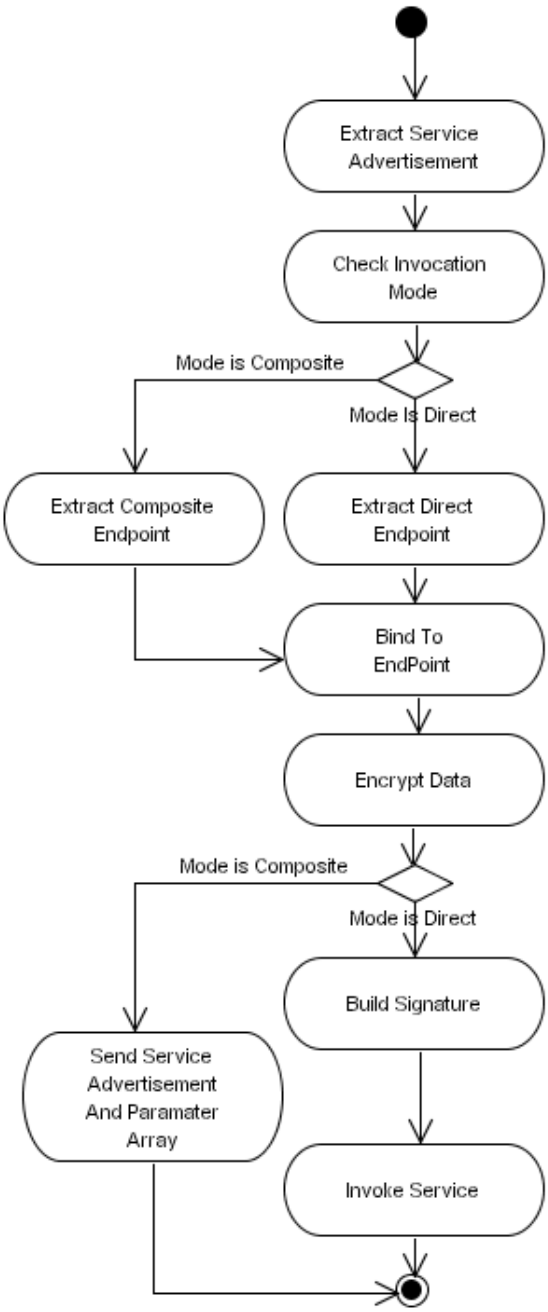

Figure C.9 Invoke Peer Service

Description:

This Activity Diagram illustrates how peer services are invoked within this framework.

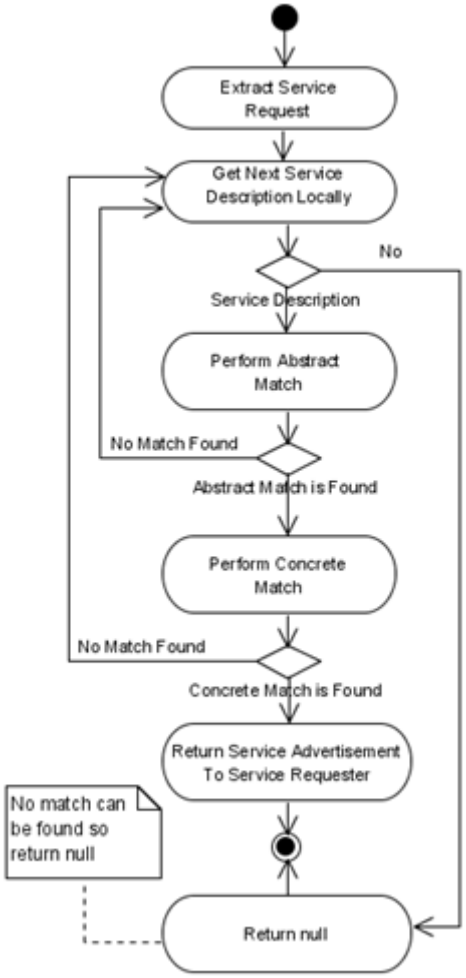

Figure C.10 Process Service Request

Description:

This Activity Diagram illustrates how service requests, received either locally or from within the P2P network are processed in this framework.

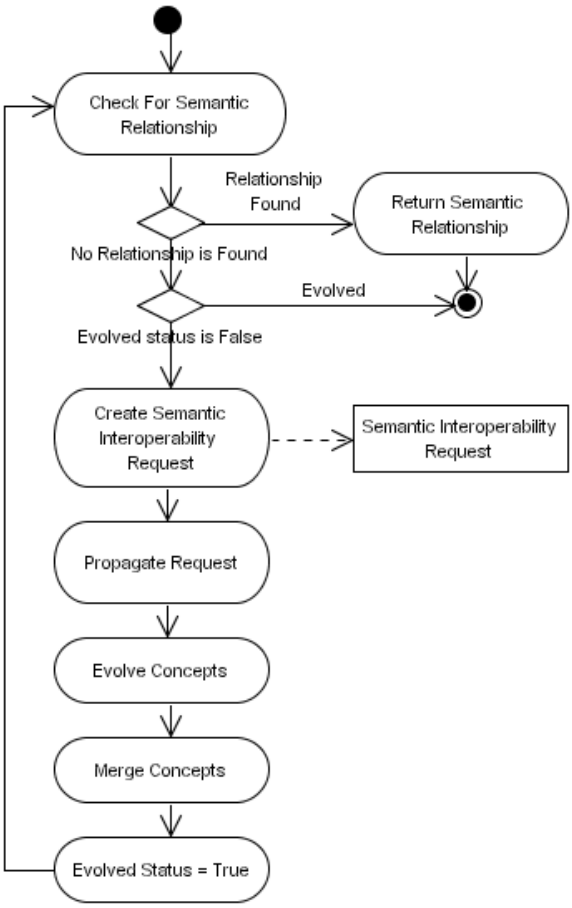

Figure C.11 Perform Semantic Interoperability

Description:

This Activity Diagram illustrates how semantic interoperability is performed within this framework.

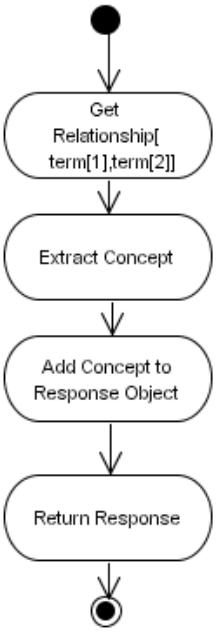

Figure C.12 Extract ontological structures

Description:

This Activity Diagram illustrates how ontological structures are extracted within this framework.

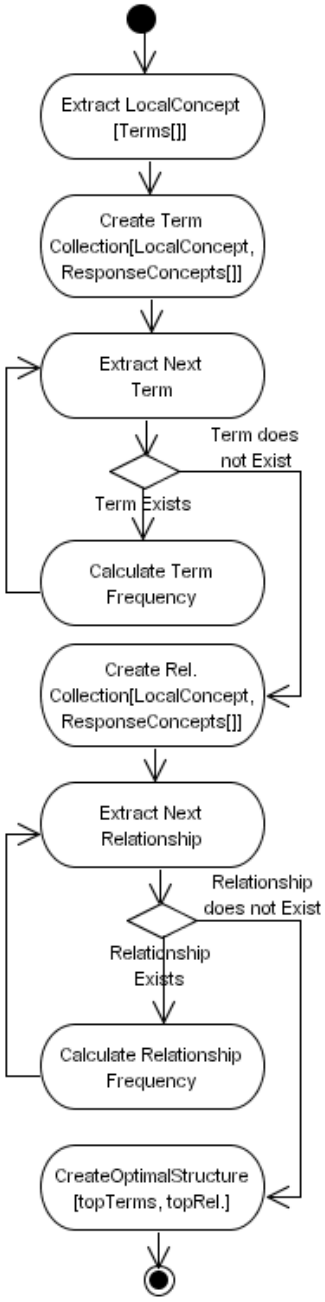

Figure C.13 Evolve ontological structures

Description:

This Activity Diagram illustrates how ontological structures are evolved within this framework.

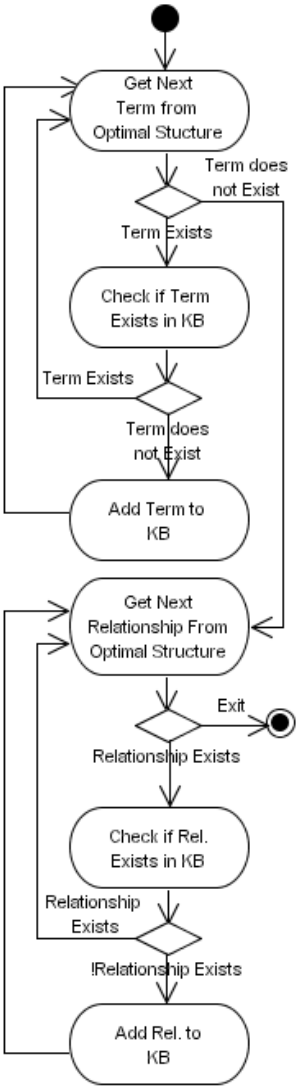

Figure C.14 Merge ontological structures

Description:

This Activity Diagram illustrates how ontological structures are merged within this framework.

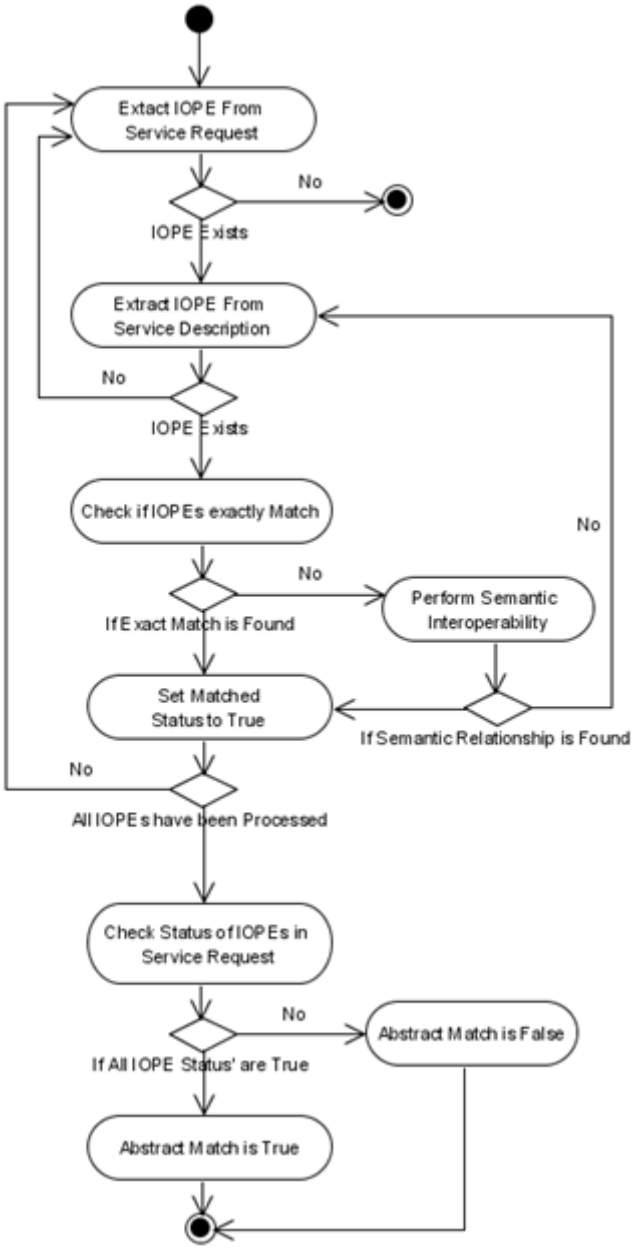

Figure C.15 Perform Abstract Match

Description:

This Activity Diagram illustrates how Abstract Matching is performed within this framework.

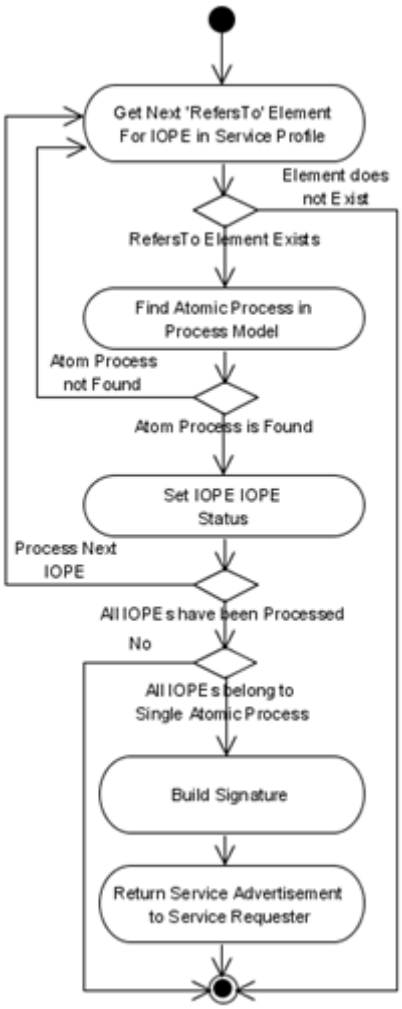

Figure C.16 Perform Concrete Match

Description:

This Activity Diagram illustrates how Concrete Matching is achieved within this framework.

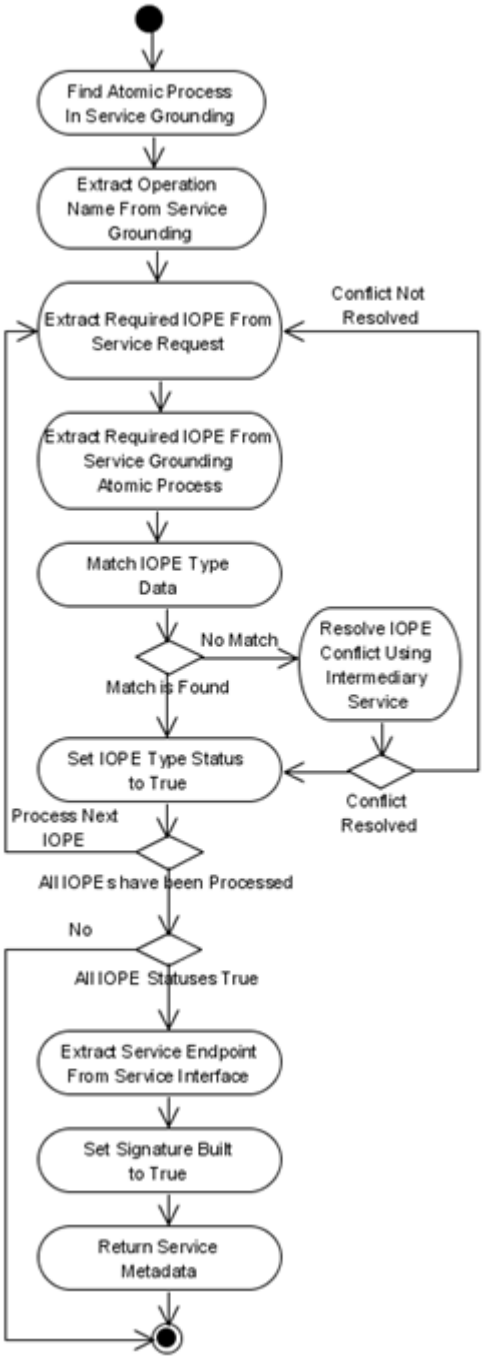

Figure C.17 Build Signature

Description:

This Activity Diagram illustrates how signatures are built within this framework.

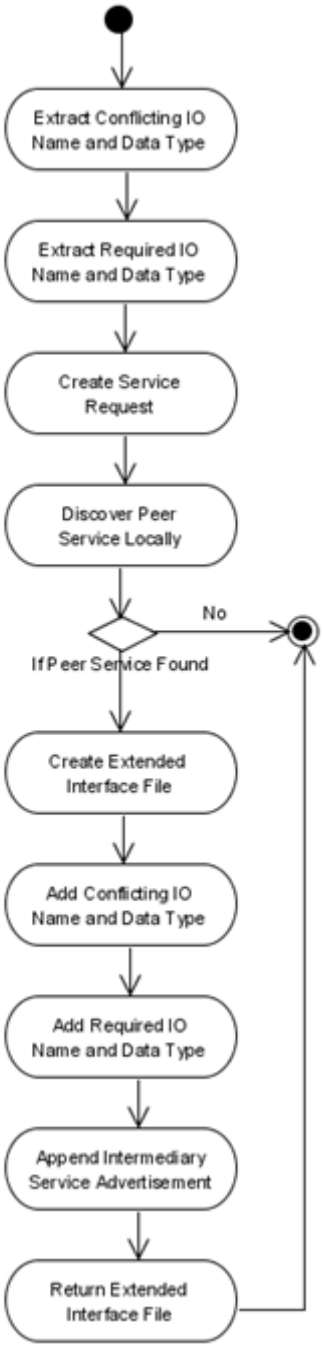

Figure C.18 Find Intermediary Service

Description:

This Activity Diagram illustrates how intermediary services are found within this framework.

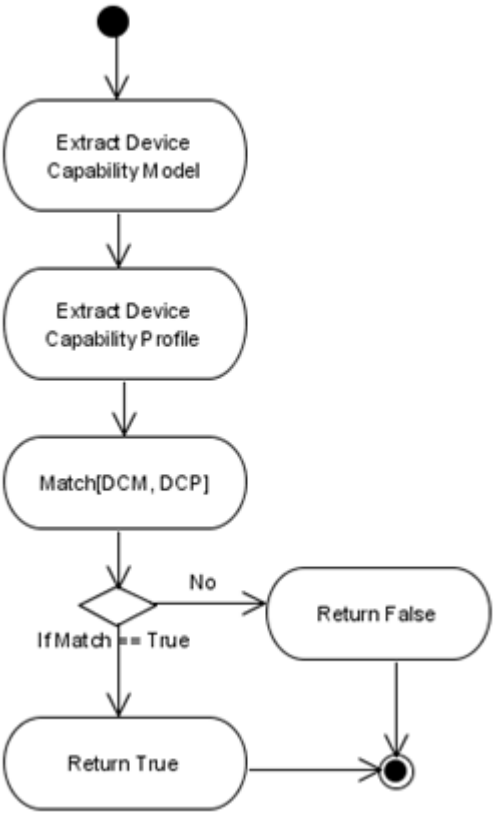

Figure C.19 Device capability matching

Description:

This Activity Diagram illustrates how device capability profiles contained in service requests are matched with device capability advertisements located on the device, within this framework.

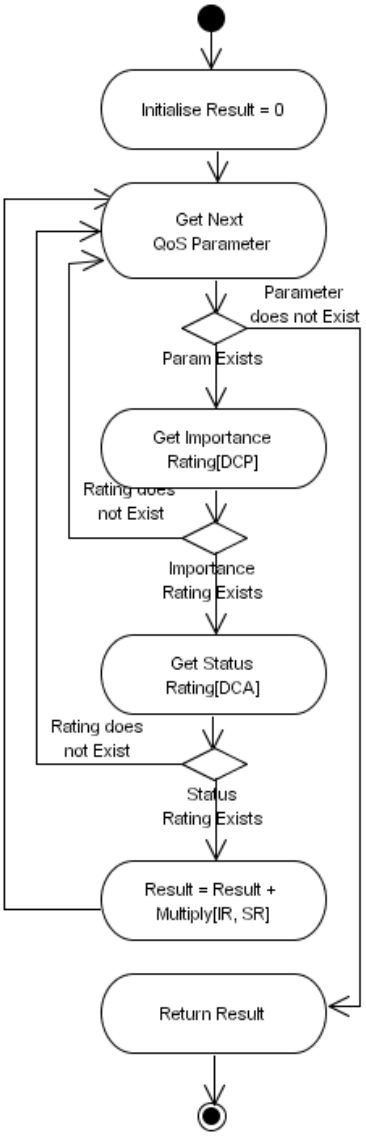

Figure C.20 Device capability matching algorithm

Description:

This Activity Diagram illustrates how the device capability matching algorithm works within this framework.

APPENDIX D: NETWORKED APPLIANCES ONTOLOGY

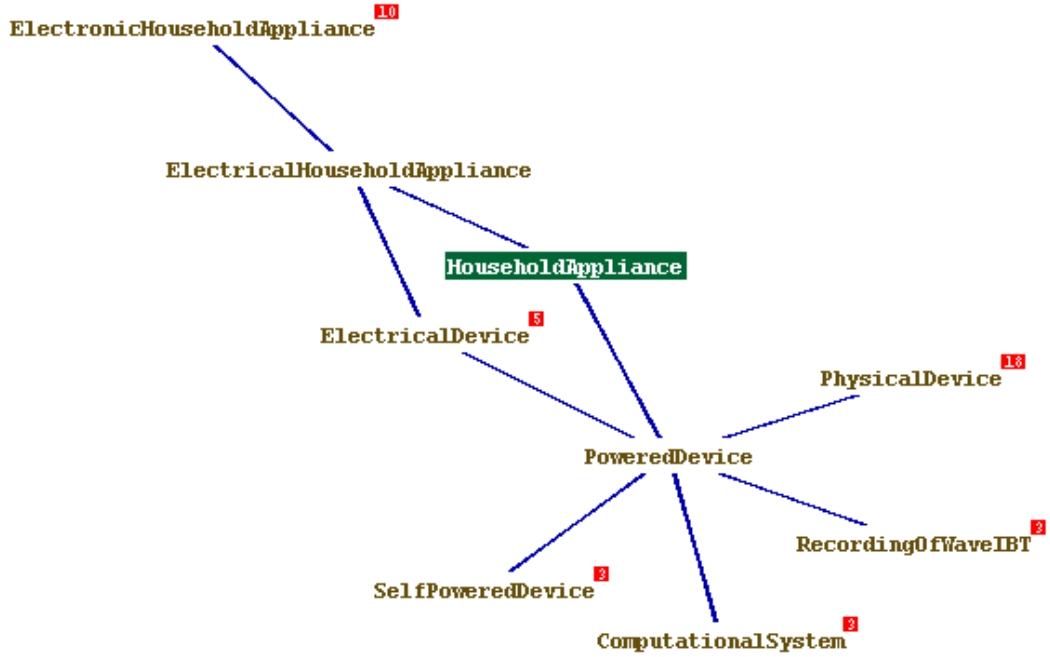

Figure D.1 Household Appliance Ontology Portion

Description:

This ontology portion describes the HouseholdAppliance concept within the DistrES ontology.

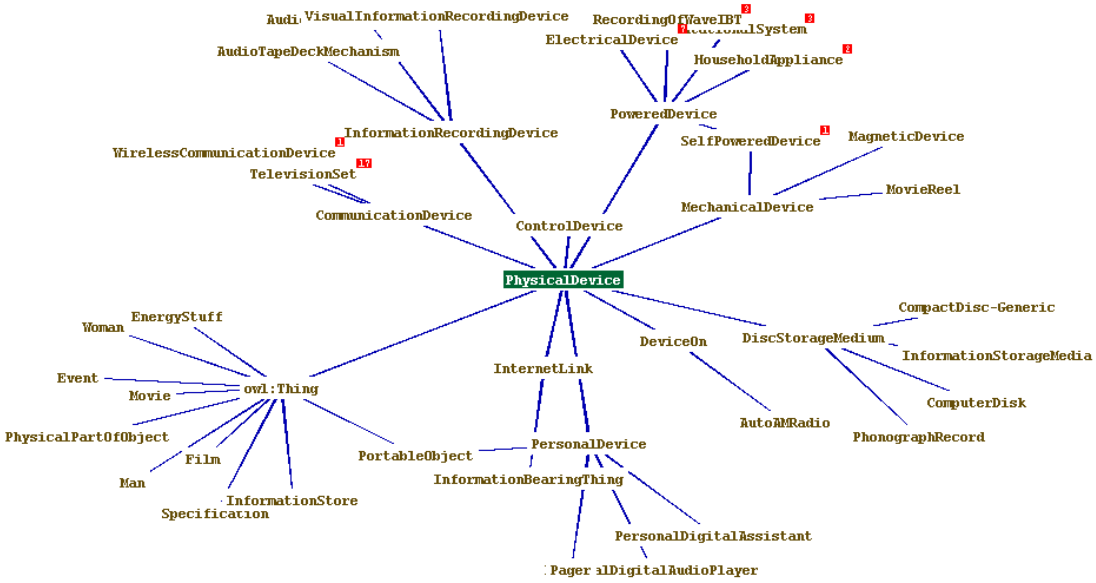

Figure D.2 Physical Device Ontology Portion

Description:

This ontology portion describes the PhysicalDevice concept within the DistrES ontology.

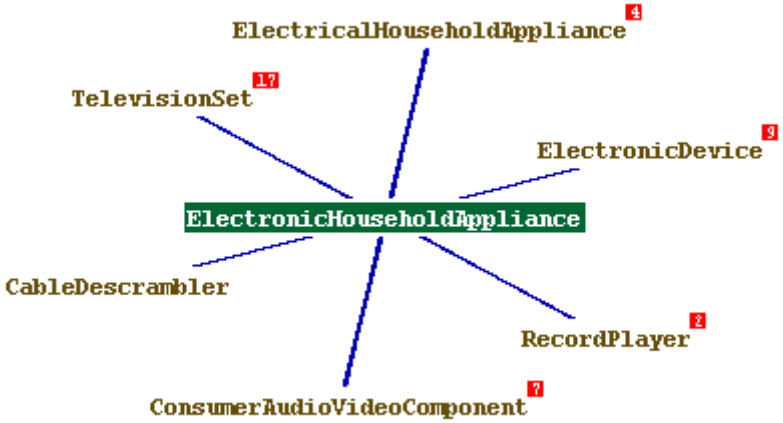

Figure D.3 Electronic Household Appliance Ontology Portion

Description:

This ontology portion describes the ElectronicHouseholdAppliance concept within the DistrES ontology.

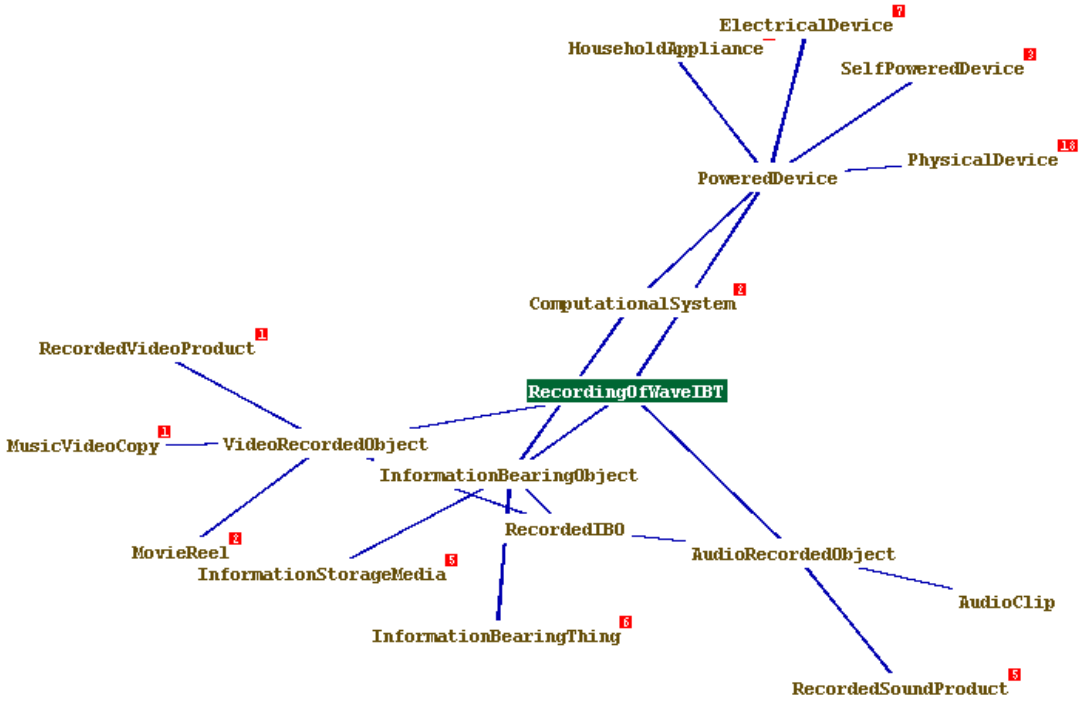

Figure D.4 Recording of Wave IBT Ontology Portion

Description:

This ontology portion describes the RecordingOfWaveIBT concept within the DistrES ontology.

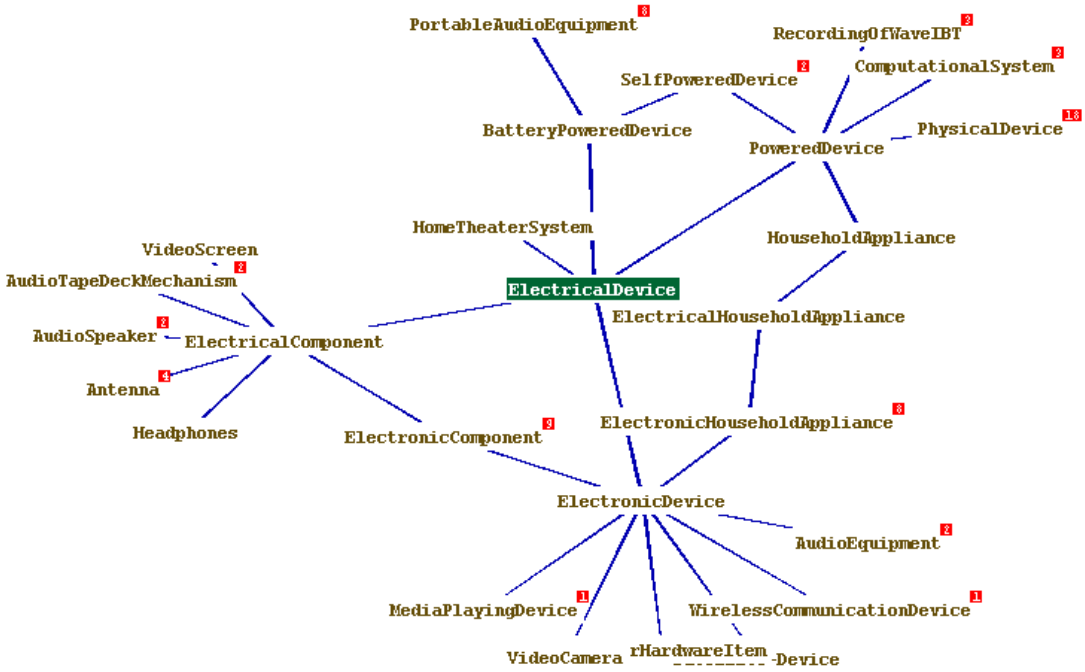

Figure D.5 Electrical Device Ontology Portion

Description:

This ontology portion describes the ElectricalDevice concept within the DistrES ontology.

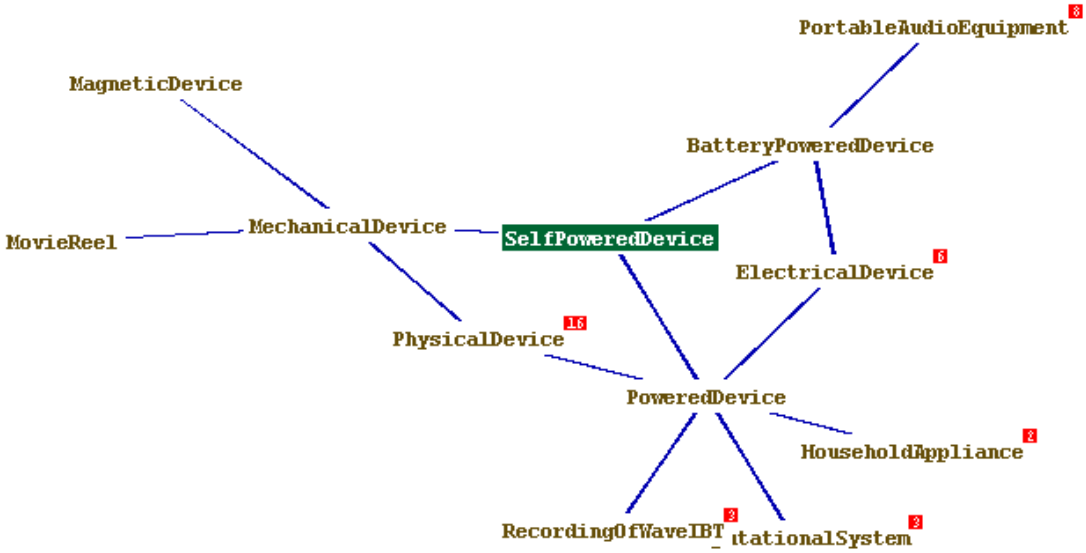

Figure D.6 Self-Powered Device Ontology Portion

Description:

This ontology portion describes the SelfPoweredDevice concept within the DistrES ontology.

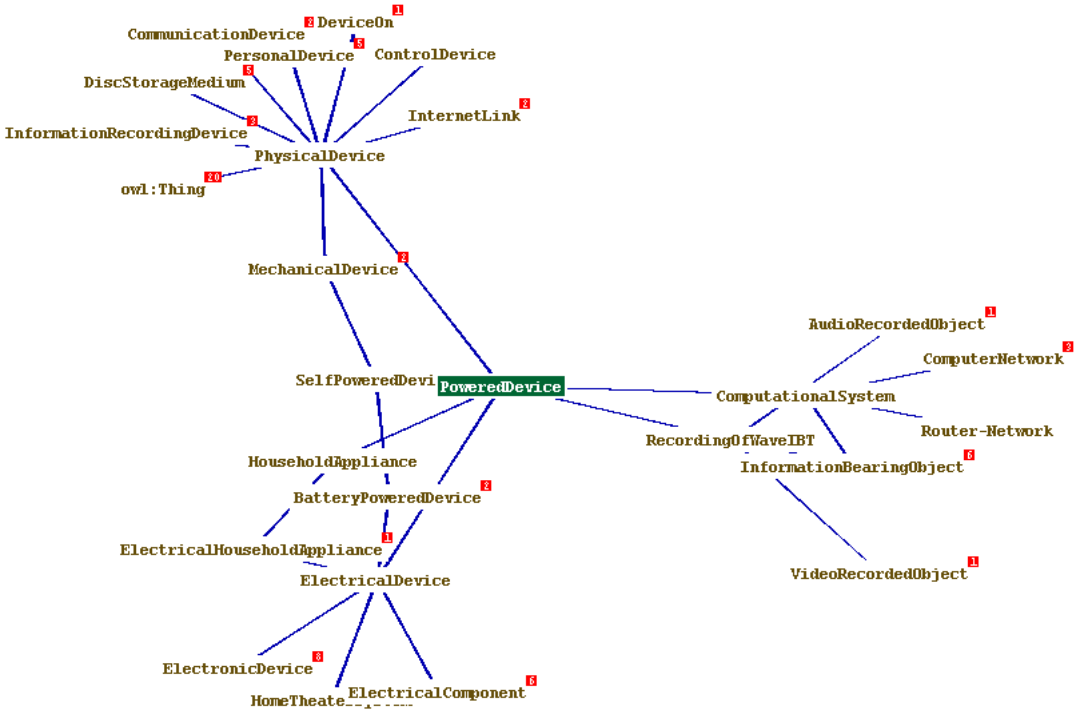

Figure D.7 Powered Device Ontology Portion

Description:

This ontology portion describes the PhysicalDevice concept within the DistrES ontology.

APPENDIX E: PUBICATIONS RESULTING FROM THIS THESIS

Merabti, M., **Fergus, P.**, Abuelma'atti, O. and Yu, H., "Networked Appliances and Home Networking," *submitted to IEEE Multimedia Magazine*, 2006.

Merabti, M., **Fergus, P.** "A Framework for Self-Adaptive Networked Appliances," *submitted to IEEE Communications Magazine: Special issue on Consumer Communications and Networking*, 2006

Fergus, P., Merabti, M., "Welcome to the Wireless Age," Review, November 2005, pp.34–35, Liverpool John Moores University, Liverpool, UK, (November 2005)

Fergus, P., Merabti, M., Hanneghan, M. B. and Taleb-Bendiab, A., "Controlling Networked Devices in Ubiquitous Computing Environments using Biofeedback," In Proceedings of *The 5th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting*, pp.91-96, Liverpool, UK, John Moores University, (June 2005).

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

Fergus, P., Mingkhwan, A., Merabti, M. and Hanneghan, M., "DiSUS: Mobile Ad Hoc Network Unstructured Services," In Proceedings of *(PWC'2003) Personal Wireless Communications*, pp.484-491, Venice, Italy, Springer, (September 2003).

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

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

Merabti, M., Abuelma'atti, O. and **Fergus, P.**, "Networked Appliances and Home Networking," In Proceedings of *The First International Workshop on the Ubiquitous Home*, Kyoto University, Japan, (March 2004).

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

Mingkhwan, A., **Fergus, P.**, Abuelma'atti, O., Merabti, M., Askwith, B. and Hanneghan, M., "Dynamic Service Composition in Home Appliance Networks," *(MTAP) Multimedia Tools and Applications: A Special Issue on Advances in Consumer Communications and Networking*, vol.31 (1), (December 2006).

Haggerty, J., Shi, Q., **Fergus, P.** and Merabti, M., "Data Authentication and Trust within Distributed Intrusion Detection System Inter-Component Communications," In Proceedings of *(EC2ND'05) 1st European Conference on Computer Network Defence*, pp.197-206, University of Glamorgan, UK, (December 2005).