A Semantic Framework for Self-Adaptive Networked Appliances

P. Fergus, M. Merabti, M. B. Hanneghan, A. Taleb-Bendiab, A. Mingkhwan

Networked Appliances Laboratory
School of Computing and Mathematical Sciences
Liverpool John Moores University,
Byrom Street, Liverpool L3 3AF, UK.
Email: {cmcppferg, M.Merabti, M.B.Hanneghan, A.Talebendiab, A.Mingkhwan}@livjm.ac.uk

Abstract: User demands and technological advances are moving us closer to the pervasive computing vision. The home of the future will include networked appliances that publish the functions they offer as intelligent 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 terms are likely. In this paper we address these issues and describe our Networked Appliance Service Utilisation Framework (NASUF), which illustrates how services offered by networked appliances can be semantically composed to extend device capabilities and perform a plethora of data adaptation functions to automatically resolve any composition conflicts that may occur. We have developed a prototype to test NASUF and our findings show that the framework can dynamically compose services provided by networked appliances in Peer-to-Peer networks.

Keywords: Networked Appliances; Interoperability; Dynamic Service Composition; Peer-to-Peer Networking; Zero-Configuration; On-Demand Service Discovery

I. INTRODUCTION

Imagine you have an Internet-enabled DVD player capable of processing MPEG-2 media formats. Using your player you try to watch a movie that is an Xvid encoding, which is a format your machine does not support. Upon discovering this unknown media type, your player tries to resolve the conflict by automatically discovering an intermediary service provided by another device within the network that can transcode the Xvid format into MPEG-2. The Xvid data is read from the disk using your player’s hardware and is sent to the intermediary service via its network connection, which transcodes the data into MPEG-2 and returns it to your player. The returned data is then streamed to your player’s MPEG-2 codec, which is processed and displayed on your TV. Such a vision provides a considerable cost saving to the consumer by allowing currently owned networked appliances to be automatically evolved to support functions they where not initially designed to do. The challenge is to make household appliances network-enabled and to publish the functions they provide as independent services. In addition intelligent middleware architectures need to be developed that automatically discover and compose these services to extend the life-time of appliances by automatically evolving device capabilities.

A number of standards such as Business Process Execution Language for Web Services (BPEL4WS) [1] and Web Service Flow Language (WSFL) [2] are attempting to compose services at a syntactical level using workflow templates, that requires a person to manually govern how this is achieved. Such standards are problematic since the service composer must decide how services are interconnected, which involves resolving any terms that are syntactically distinct but semantically similar; checking that parameter data types are compatible; and ensuring that services are correctly invoked.

Other research initiatives such as Universal Plug and Play (UPnP) [3], Open Services Gateway Initiative (OSGi) [4], Home Electronic System (HES) [5], Home Audio and Video Interoperability (HAVi) [6] and Digital Home Working Group (DHWG) [7] have also tried to standardise how devices are interconnected by describing and discovering services using attribute-value pairs which are known a priori. These standards do not provide any mechanisms to describe and discover services semantically; consequently it is very difficult to automatically compose services in ad hoc environments without any human intervention.

Furthermore the above standards fail to address interoperability between open standards and the vocabularies used to discover and describe devices and the services they provide. 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 performs semantic interoperability between different vocabularies using ontologies. However the major difficulties that still need to be addressed are how ontological structures are distributed, managed and evolved over time, within P2P networks, based on general consensus [12].

In this paper we illustrate how services can be described semantically and automatically composed using high level descriptions, which relate to the “what” part of the service composition rather than the “how”. We argue that using ontologies in this way is a key requirement if services are to be
discovered and automatically composed devoid of any human intervention. We present a framework we have developed called the Networked Appliance Service Utilisation Framework (NASUF), which allows the services provided by devices to be automatically composed to produce value-added services and perform a range of data adaptation functions to automatically resolve any composition conflicts that may occur. Combining the advances made in Peer-to-Peer networking, ontology, semantic web services and signature matching, forms the basis of our approach. We begin in Section 2 by providing an overview of NASUF. In Section 3 we describe how our Semantic Interoperability and Signature Matching (SISM) Service works and in Section 4 we describe our implementation before providing a conclusion and describing our future work in Section 5.

II. NETWORKED APPLIANCE SERVICE UTILISATION FRAMEWORK OVERVIEW

In our previous work we have successfully developed several components that allow us to host and discover unstructured services [11]; evolve knowledge structures to enable semantic interoperability between different vocabularies [12]; and publish the functions offered by complex devices as individual services [13]. The NASUF framework is illustrated in Figure 1 and the components it uses are described below.

NASUF PEER

- **DiSUS** is a component we developed that allows devices to host and discover unstructured services in P2P networks, devoid of any centralisation [11].
- The DistrES Service we developed allows ontological structures to be evolved within a P2P network based on general consensus and resolve terminology differences between concepts that are syntactically distinct but semantically equivalent [12].
- We developed the Device Capability Service to determine if the device providing the service has the required hardware, software and network capabilities to effectively execute it.
- The Context Service processes ontological structures, which describe the contexts in which the device can be used. Contexts are dynamically evolved over time using the DistrES service and are used to guide automatic service composition. They describe high-level descriptions that relate to the “what” part of the service composition.
- The SISM Service forms the main part of this paper and performs dynamic service composition between service-enabled devices in a P2P network based on device and service capability matching.
- Devices will support application specific peer services (PS), which expose the functionality the device provides as individual services. Peer services provide a level of abstraction that map to any implementation service technology used.

In this paper we extend this work and include mechanisms that allow devices to automatically discover and compose the services they provide using semantic descriptions. Thus providing a self-adaptive service composition mechanism engenders further technical challenges, which we describe below:

- 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 that a high quality of service is maintained.

In the remainder of this paper we describe how these challenges are addressed using the SISM Service we have developed.

III. THE SEMANTIC INTEROPERABILITY AND SIGNATURE MATCHING (SISM) SERVICE

SISM is a semantic matching service that determines if service descriptions hosted by a device match service requests it receives from the P2P network. This is achieved by processing the 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 they describe, which are more commonly referred to as IOPEs [8]. Using IOPEs provides the basis for signature matching within the SISM Service and is described in more detail in the following sections.

A. The IOPE Matching Process

Using the SISM Service we 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 2. In this instance 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 a relationship cannot be found, the unknown term is mapped into the DistrES ontology [12]. 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.

When abstract matches are found, SISM retrieves the service ontologies [8], along with the service interface file and creates a table containing the matching IOPEs from the service request. 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.

### B. 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’s interface file by processing all the service ontologies. SISM processes the Service Profile [8] and retrieves the values associated with the ‘refersTo’ element. These values specify which ‘Atomic Process’ [8] 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 file 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 file, 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, which 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.

- 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.
- 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.
- The ‘Atomic Process’ in the service process model for ‘Service A’ has an associated output element that

---

Fig. 2 Illustrates IOPEs matching

- 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 request subsumes the IOPE in the service request then this constitutes a plug-in match [9].
- If the IOPE in the service request subsumes the IOPE in the service description then this constitutes a subsumption relationship [9].
- Anything else fails.

If a relationship cannot be found, the unknown term is passed to the DistrES Service [12] and propagated within the P2P network. This results in zero or more semantic structures being returned that describe how the term is defined – using evolutionary programming techniques the structures are evolved until an optimal solution has been produced and merged into the DistrES ontology [12]. Once the structure has
conceptually matches an output described in the service request.

- 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.

**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.

- 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.

- 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.

- 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’.

- Anything else fails.

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

**IOPE Conflict Resolution**

One of the main challenges is to form compositions with other services irrespective of whether the service can directly accommodate a one-to-one composition as illustrated in Figure 3.

In this example “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.

![Fig. 3 Illustrates dynamic service composition](image)

Finding an intermediary service is achieved by propagating a reformulated service request to the P2P 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 File (EIMF). The EIMF 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, which are defined in the service request, will be added to the EIMF. For example a typical home environment may provide several visual display services capable of processing streamed data – typically you will select devices that will provide the best quality of service, 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 film. However, if the mobile phone is the only device available, then an intermediary service will be discovered to transcode the DVD data into a format that can be readily processed by the mobile phone to ensure a high quality of service is maintained. As a result each device that joins the network within our framework must describe and publish the hardware, software and networking capabilities it supports.

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

**C. The Extended Interface (EI) service**

The EI service, as illustrated in Figure 4, is invoked when a service provided by the device does not directly support a method invocation.

![Fig. 4 Extended Interfaces for the Visual service](image)

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

The EIMF 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. If the connection mode is ‘composite’ then the EI service processes the EIMF for the intermediary service it needs to use before connecting to its EI service and passing it the metadata and the parameters. This process continues until a direct connection with a service in the composition is made as illustrated in Figure 4.

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 was implemented.

**IV. Implementation**

The prototype we developed reads audio and video (AV) data from a file and streams it between JXTA peers [14]. It does not explicitly perform the transcoding functions, however the prototype illustrates how the SISM service resolves data type conflicts using semantic metadata to describe service requests and service descriptions. The case study has the provision to provide, a TV and two DVD Players. The TV provides a “Visual” service; “DVD Player 1” provides a player and a “MPEG Codec” service including a “Movie” service. “DVD Player 2” provides the same services as “DVD Player 1” except it has an “AV-to-MPEG” service instead of an “MPEG Codec” service. The case study is illustrated in Figure 5.

“DVD Player 1” processes the Movie Service Description and determines that the “MPEG Codec” cannot process the AV media format it describes. The SISM algorithm resolves this conflict by creating a service request, which defines two IOPEs – the first IOPE is the conflicting Input (AV) and the second IOPE is the required output needed to resolve the conflict (MPEG). The service request is propagated within the P2P network, which is processed by “DVD Player 2”. “DVD Player 2” determines that the IOPEs can be matched at an abstract and concrete level and creates an extended interface metadata file. This file is returned to “DVD Player 1”, which is stored as an intermediary service that can transcode the AV media format into MPEG.

Once the data type conflict has been resolved “DVD Player 1” tries to discover a “Visual” service capable of processing the data outputted from the “MPEG Codec” service. The TV receives the service request and determines that it can process the video and audio data using the “Visual” service it provides. The TV returns a service description to “DVD Player 1”, which describes how the “Visual” service can be invoked. When the composition is executed, “DVD Player 1” connects to “DVD Player 2” and streams the AV data. “DVD Player 2” transcodes the data using the “AV-to-MPEG” service and streams the results back to “DVD Player 1”, which is processed by the MPEG codec and in turn streamed to the “Visual” service.
The TV and DVD Players have JXTA interfaces that allow them to join the default peer group called ‘NetPeerGroup’ [14]. Streamed data is sent between devices in the P2P network using JxtaSocket/JxtaServerSocket objects and Pipe Advertisements [14]. When devices discover an intermediary service they extract the Pipe Advertisement from the service description and use it to bind to the services JxtaServerSocket. Data is streamed to the JxtaServerSocket, which is transcoded and returned to the service requester. All services where developed using Java and service requests and service descriptions were described using OWL-S [8]. Semantic interoperability between terms was achieved using the DistrES ontology we developed. The OWL-S service ontologies and the DistrES [12] ontology where queried using the RDF Query Language (RDQL) API provided by the Jena 2.0 API toolset [15].

V. CONCLUSIONS AND FUTURE WORK

In this paper we argue that existing service-oriented middleware architectures do not provide mechanisms that allow services to be automatically discovered and composed in ad hoc network environments without any human intervention. Consequently the ability to extend the functions provided by devices using self-adaptation is not possible. We provide an overview of our NASUF framework and illustrate how it enables self-adaptive service-oriented networked appliances to evolve the functions they provide by automatically discovering and composing services in a P2P network without having to know the service interfaces a priori. We have illustrated how we grounded our ideas using a prototype that describes how the SISM service was implemented. The paper illustrates that services can be automatically discovered and composed in P2P networks without any human intervention. In our future work we will address the exponential problem associated with discovering and composing intermediary services. Furthermore we will also replace the EIMF with a Service Process Model [8] in order to improve the performance and we will enhance the SISM Service to resolve multiple IOPE conflicts that may occur in signatures.

REFERENCES


Fig. 5 Automatic Media Format Transcoding