UNIVERSITY
of
GLASGOW

Application Collaboration in Ubiquitous Computing Environments

Steven Samuel Heeps

Department of Computing Science
University of Glasgow

A dissertation submitted in fulfilment of the requirements for the degree of Master of Science by Research at the University of Glasgow

Feb 1, 2008

Abstract

With the emergence of mobile and ubiquitous computing environments, there is a requirement to enable collaborative applications between components of these environments. As many of these applications (e.g. MP3 players) have been designed to operate in isolation, making them work together is often complicated by two, different aspects: firstly, a lack of protocols to enable the systems to bind to each other for interaction and, secondly, semantic and ontological differences in the meta-data describing the data to be shared. An abstraction termed a Self-Managed Cell has previously been proposed as an architectural pattern for building autonomous systems, that can represent entities ranging from individual devices to entire environments, and have described mechanisms that enable such cells to establish peer-to-peer bindings facilitating interaction at the system and management level. Semantic and ontological differences in the meta-data describing information to be shared between peers and application level aspects of interaction still exist, and prevent successful, autonomous application collaboration.

Typical approaches to application collaboration, particularly in the database world, require the presence of a third-party administrator to manage ontological differences; such an approach is incompatible with interactive, autonomous systems. This dissertation presents a novel approach to automatic collection mapping suitable for deployment in autonomous, interacting systems. The approach facilitates the collaboration of SMC application-level data collections by identifying areas of conflict and using meta-data values associated with those collections to establish commonality. Music sharing and traditional "book" library catalogue matching applications, exploiting this mapping mechanism, have been developed to facilitate the sharing of data between peers. Protocols and abstractions are used to establish commonality and collaboration between the systems, and the mapping mechanism is used to enhance interoperability at the application level.

Declaration

The work in this dissertation is based on research carried out at the Embedded, Networked and Distributed Systems research group, the Department of Computing Science, University of Glasgow, Scotland.

The work described herein is part of a larger project, AMUSe (The Autonomic Management of Ubiquitous Systems for e-Health), which was a joint collaboration between Department of Computing Science, University of Glasgow and the Department of Computing, Imperial College London. The dissertation will describe work which I have undertaken in collaboration with other project members together with work for which I have been solely responsible.

Copyright © 2008 by Steven Samuel Heeps.

Acknowledgements

The author wishes to thank the UK Engineering and Physical Sciences Research Council for their support through grants GR/S68040/01, GR/S68033/01 and GR/N15986/01.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Recent advances in ubiquitous and mobile computing have dramatically changed the role of the computer in users' lives and made mobile computing the new personal computing and communication paradigm. The overriding motivation is that computing systems should seamlessly integrate into the life of the user and interoperate with other systems to offer mobile services as and when desired. A particular challenge is the ability to manage the services, information and integration of such ubiquitous systems.

Existing network and system management frameworks do not adequately cater for ubiquitous computing environments. Current frameworks are predominantly aimed at large-scale corporate environments, telecommunications networks and internet service providers [1]. To realise autonomic management in ubiquitous systems, it is necessary to define an architecture which scales down to small, lightweight structures with local decision making capability.

1.1.1 The Self-Managed Cell

The Self-Managed Cell (SMC) is a fundamental management design pattern for autonomous systems [1]; the SMC architecture provides policy-based, autonomic management capabilities for ubiquitous computing environments [2–5]. An SMC consists of hardware and software components that form a cell capable of functioning autonomously. The minimum requirements for the cell include functionality for measurement and event correlation, policy-based control mechanisms, a discovery service and management components. The AMUSe project chose to validate the general SMC concepts in e-Health environments where mobile patients can be monitored by multiple devices as they participate in activities of daily living; such devices should form an adaptive, auto-configured body-area network,

not requiring administration by non-technical patients or medical staff. This dissertation will apply insights gained from e-health application development to other ubiquitous environments; this dissertation will focus on music sharing book library catalogue matching applications.

The Self-Managed Cell architecture is more comprehensively discussed in Chapter 2.

1.1.2 Application Collaboration

Enabling multiple SMCs to collaborate is essential in providing network services and distributed information.

In ubiquitous computing environments, SMCs need to collaborate without having a pre-agreed schema, and it is also desirable that there is agreement and common semantics for applications and devices. The ultimate vision is that SMCs can interact without requiring administration by non-technical users. This is challenging, as it takes autonomic computing beyond traditional network management where systems integration is typically handled centrally and relies heavily on a human administrator to perform network management across entire corporate systems. The SMC architecture currently supports integration at the system and management level where the basics for SMC interaction are handled in terms of policy, data and event exchanges [6]. Successful SMC integration at this level provides the mechanisms for services at the application level to collaborate.

This dissertation explores the challenge of integration at the application level. Semantic differences between collaborating applications are usually managed by an administrator who maps the differences or documents a strict ontology to which systems developers and users adhere. It is likely that ontological and semantic differences between individual applications will prove a barrier to application collaboration due to the autonomous nature of the environment. Such differences may not be evident or challenging in e-health environments due to the closely controlled, closed nature of healthcare systems. It is likely that applications collaborating in healthcare environments will have been developed by the same provider or at least follow pre-defined schematic guidelines as previously mentioned.

This dissertation will present an automatic ontology generation and mapping mechanism to overcome such conflicts. A music sharing implementation is described which integrates the ontology generation and mapping mechanism with the SMC architecture [7, 8]. Further research has also been under taken to assess the ability of the collaboration mechanism, described herein, on matching different library catalogue records.

1.2 Thesis Statement

I assert that it is possible for independent, ubiquitous systems to dynamically collaborate at the application level, particularly where semantic conflict in the application data exists. A successful application collaboration should utilise mechanisms provided at the system and management level to facilitate integration, and provide mechanisms to overcome semantic and ontological differences between systems at the application level, specifically applications defined by data collections characterised by meta-data. An implicit assumption is that such applications possess only a partially shared ontology.

I shall demonstrate the validity of this assertion by defining and developing exemplar application-level services capable of running on a Self-Managed Cell, describing a suitable protocol to facilitate collaboration at the application-level, implementing this protocol, and demonstrating the effectiveness of the collaboration through multiple real-world prototypes.

1.3 AMUSe

As previously mentioned, some of the work described in this dissertation is part of a larger project named AMUSe (Autonomic Management of Ubiquitous Systems for e-Health). The work of AMUSe is described in detail in the following papers, two of which I am first author and the remainder co-author.

- S. Strowes, N. Dulay, S. Heeps, S. Keoh, E. Lupu, A. E. Schaeffer-Filho, M. Sloman and J. Sventek, Wide-Area SMC Interaction, Implementation and Emulation, University of Glasgow Department of Computing Science Technical Report.

- S. Heeps, J. Sventek, N. Dulay, A. Schaeffer-Filho, E. Lupu, M. Sloman and S. Strowes, Dynamic Ontology Mapping for Interacting Autonomous Systems, Proc. IEEE Int. Workshop on Self-Organizing Systems, Lancaster, United Kingdom, September 2007, Springer Vol 4725/2007, pp. 255-263.

- S.-L. Keoh, N. Dulay, E. Lupu, K. Twidle, A. Schaeffer-Filho, M. Sloman, S. Heeps, S. Strowes and J. Sventek, Self-Managed Cell: A Middleware for Managing Body-Sensor Networks, Proc. Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services, Philadelphia, USA, August 2007.

- A. Schaeffer-Filho, E. Lupu, N. Dulay, S.-L. Keoh, K. Twidle, M. Sloman, S. Heeps, S. Strowes and J. Sventek, Towards Supporting Interactions between Self-Managed

Cells, Proc. IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems, Boston, USA, July 2007, pp. 224-236.

- E. Lupu, N. Dulay, J. Sventek and M. Sloman Autonomous Pervasive Systems and the Policy Challenges of a Small World Proc. 8th IEEE Int. Workshop on Policies for Distributed Systems and Networks (Policy 2007), Bologna, Italy, June 2007, pp. 3-7. (invited)

- A. E. Schaeffer Filho and E. Lupu Abstractions to Support Interactions Between Self-Managed Proc. 1st Int. Conf. Autonomous Infrastructure, Management and Security, (AIMS 2007) Doct. Sypm. , Oslo, Norway, June 21-22, 2007, pp.160-163.

- E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, K. Twidle, S.-L. Keoh and A. Schaeffer-Filho, AMUSE: autonomic management of ubiquitous e-Health systems, Concurrency and Computation: Practice and Experience, ISSN: 1532-0634, DOI:10.1002/cpe.1194, May 2007.

- S. Keoh, K. Twidle, N. Pryce, A. Schaeffer-Filho, E. Lupu, N. Dulay, M. Sloman, S. Heeps, S. Strowes, J. Sventek and E. Katsiri, Policy-based Management of Body-Sensor Networks, Proc. Int. Workshop on Wearable and Implantable Body Sensor Networks, Aachen, Germany, March 2007, Springer pp 92-98.

- S. Strowes, N. Badr, N. Dulay, S. Heeps, E. Lupu, M. Sloman and J. Sventek, An Event Service Supporting Autonomic Management of Ubiquitous Systems for e-Health, Proc. Int. Workshop on Distributed Event-Based Sytems, Lisbon, Portugal, July 2006, p22-27.

- S. Heeps, N. Dulay, A.E. Schaeffer, E. Lupu, M. Sloman, S. Strowes, J. Sventek, The Autonomic Management of Ubiquitous Systems Meets The Semantic Web Proc. 2nd Int. Workshop on Semantic Web Technology For Ubiquitous and Mobile Applications (SWUMA'06), Trentino, Italy, August 2006.

- N. Dulay, E. Lupu, M. Sloman, J. Sventek, N. Badr and S. Heeps, Self-Managed Cells for Ubiquitous Systems, Proc. 3rd Int. Conf on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS 2005), St Petersburg, Russia, LNCS 3685, pp. 16, Sept 2005. (invited)

- N. Dulay, S. Heeps, E. Lupu, R. Mathur, O. Sharma, M. Sloman and J. Sventek, AMUSE: Autonomic Management of Ubiquitous e-Health Systems, Proc. UK e-Science All Hands Meeting, Nottingham, UK, September 2005.

- J. Sventek, N. Badr, N. Dulay, S. Heeps, E. Lupu and M. Sloman, Self-Managed Cells and their Federation, Workshop Proceedings of the 17th Conf. on Advanced Information Systems Engineering (CAiSE05), pp. 97-107, June 2005 (invited).

For clarity, my primary contribution with respect to AMUSe is in the development of application level mapping mechanisms described herein. Additionally I have been involved in design, implementation and review of a number of aspects related to the Self-Managed Cell, the fundamental management architecture defined by AMUSe and documented in Section 2.2. I have participated in the design and implementation of the Discovery and Event Service, together with undertaking AMUSe scenario development.

1.4 Dissertation Organisation

The main body of the dissertation begins in Chapter 2 where the background to the dissertation is described and the work comprising the larger project, AMUSe, which describes the Self-Managed Cell concepts upon which this dissertation builds is introduced. Chapter 3 introduces motivating scenarios which serve to place the work and highlight situations where solutions proposed herein can be applied. A survey of related work is presented in Chapter 4, discussing current approaches to solving related problems. Chapter 5 introduces the system designed and identifies how this attempts to address the problems defined in the motivating scenarios presented in Chapter 3. The implementation is described in Chapter 6. Measurements and evaluation are presented in Chapter 7. Chapter 8 concludes and covers areas of future potential work.

Chapter 2

AMUSe and The Self-Managed Cell

This dissertation describes work which attempts to utilise the fundamental architectural concepts developed as part of the AMUSe project. Therefore this chapter will detail AMUSe and the Self-Managed Cell (SMC) Architecture which will be referenced throughout and will attempt to place the ontology generation and mapping mechanism in relation to the SMC architecture.

Related work, in respect to the work of others, is described in detail in Chapter 4.

2.1 AMUSe

The European Commission have set ambitious e-Health targets for member states to achieve in the coming years. In particular, member states should develop Health Information Networks, accelerating information flow through health care systems. e-Health applications of the future will involve mobile patients interacting with a ubiquitous computing environment that detects their activity, current context and adapts accordingly. However, the promise of such ubiquitous computing environments will not be realised unless these systems can effectively disappear. Developing the architectures, tools and techniques that permit such environments to become self-managing is therefore essential.

AMUSe focused on the architecture and development of autonomous management capabilities for ubiquitous computing environments, in general, and e-health environments, in particular. AMUSe advocated the concept of the Self-Managed Cell (SMC) and in 4 working packages developed an SMC, created methods for inter-cell composition, Cell federation and layering whilst developing various demonstrators and evaluators. This dissertation focuses, generally, on the SMC architecture and specifically on mechanisms

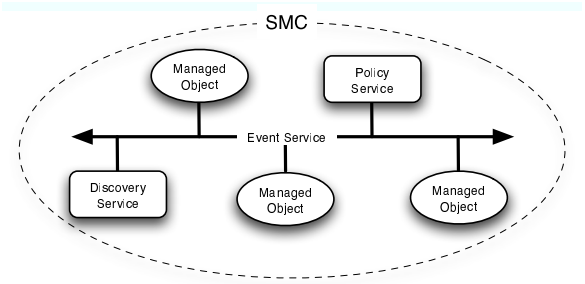

Figure 2.1: Self-Managed Cell Architecture

for application integration over the SMC architectures.

2.2 The Self-Managed Cell

An SMC manages a ubiquitous computing environment comprised of a number of devices and services. Devices may, for example, include smart phones, music players, PDAs or on-body sensors which form part of a body-area network. An SMC must support autonomic functions such as self-configuration, self-healing, self-optimisation, self-protection and self-awareness [9]. In a typical e-Health scenario, patients or medics will not typically have the technical knowledge to configure sensors on a body-area network. Instead an SMC should automatically discover appropriate sensors and configure them accordingly. As shown in Figure 2.1, there are several core services which constitute an SMC; a policy service, a discovery service and an event service.

Policies provide a means of specifying adaptive behaviour in systems management. Hence, policies are the rules that govern the behaviour of an SMC. Ponder2[1] is the policy service for the SMC and is a scalable, extensible policy framework suitable for limited resource devices. Its development was based upon previous policy definition experience [10]. Ponder2 comprises a general purpose object management system, a domain service, obligation policy interpreter and a command interpreter. The Domain Service provides a hierarchical structure for managing SMC objects. The Obligation Policy Interpreter handles Event-Condition-Action rules. The Command Interpreter accepts a set of commands via several communication interfaces. These commands can be used to interrogate the Domain Service and perform invocations on the managed objects. Managed Objects are

[1] http://ponder2.net/

tangible entities, usually a physical device such as a PDA or on-body sensor, which represent SMC devices, services within those devices and remote SMCs. Each device has a unique identifier, such as IEEE 802.* MAC address. Domains and policies are also managed objects in their own right upon which actions can be performed, for example adding/removing an object from a domain or enabling/disabling a policy. The overall architecture of the policy service comprises the domain structure, the triggering mechanism matching events to obligation policies and the execution invocation engine which is used to make calls to the objects inside the domain structure. The policy service has an event interface through which event notifications are received from the SMC event bus, an invocation interface through which external invocations are received and an action interface for invocations on external objects [3].

The discovery service is responsible for managing SMC group membership. It handles the admission of new services when they enter communication range (employing authentication if required) and the removal of services that have left the SMC (perhaps through physical removal or a discharged battery). While the discovery protocol uses other forms of network messaging to determine when components arrive or depart, it only uses the event service to notify other components of the SMC about device arrival/departure. When devices arrive or depart an SMC, the discovery service notifies other components of the SMC through New Member and Purge Member events, respectively [2].

The event service is a content-based publish/subscribe bus over which SMC services send management traffic. It delivers event notifications generated by services within the SMC to components which have registered an interest in receiving such events. The event service forms the backbone of every SMC and is used by all management services to communicate, although consenting components/services are able to use other communication mechanisms for non-management traffic. All communication between services and the event service are synchronous; services know that the event service has acknowledged their new events, and the event service knows that other services have received an outgoing event. If a subscriber does not acknowledge an event, that event will be queued for later delivery. To allow the SMC to use various devices, the event service communicates to each member of the SMC via that members own proxy. It is not expected that the event service will have to deal with high volumes of events; indeed, the target platform type is a PDA, which places constraints on the memory footprint. Thus, the event service must be lightweight while providing the necessary functionality [2].

As an example, an SMC may represent a patient body-sensor network consisting of a temperature sensor and alarm. Initially the discovery service detects the sensor as a

new device and is added to the appropriate domain within the policy service. A policy is defined stating that an alarm should be raised when the patient temperature exceeds a defined threshold. The obligation policy interpreter then monitors the event service for a temperature exceeded event and on seeing this invokes a raise alarm action. This example highlights a cell of minimal complexity, however patient monitoring cells will typically consist of multiple devices, for example glucose, blood pressure, heart-rate and ECG monitors, and tens of polices which are self-managed via the policy system.

2.3 Self-Managed Cell Composition

SMC collaboration is essential in supporting peer-to-peer interactions between cells in order to collaborate and share devices and services; for example, two peers automatically discovering and sharing music on a train, or a doctor seamlessly accessing health information provided by a patient's body-area network. For such examples to become a reality, an SMC must know what kind of interfaces its neighbours support and what kind of protocols or commands they understand. In a truly ubiquitous scenario it may not be reasonable to assume such knowledge.

SMC collaboration may occur at two levels, the SMC system and management level and the application level. Figure 2.2 highlights the typical SMC level architecture for peer-to-peer collaboration. Initial interaction establishment between SMCs is handled at system and management level. Peers are able to invoke actions and load policies onto others whilst retaining their autonomy. The discovery service initiates the peer-to-peer interaction by discovering another SMC and automatically decides, via policies, the type of interaction desired and bootstraps that interaction. A proxy is generated which enables event propagation between peers. Events announce the discovery of a new SMC, obligation policies determine the mode of interaction between peers and interfaces export different functionalities derived from the discovered peer's profile. Once an interaction has been established peers can begin to collaborate. Collaboration is supported by the concept of a mission. Missions define the behaviour of the interaction between two SMCs and are composed of a group of policies which define the duties of the peer SMC, in terms of the obligation policies it should enforce. Roles are also an important aspect of successful collaboration. Roles are placeholders within the local SMC domain structure for SMCs yet to be discovered. They define the expected operations that a remote SMCs must provide and represent the minimum requirements an SMC of a given profile must comply. Policies can then be written and applied in terms of roles. Ultimately, when an SMC is discovered it is assigned a role, and obligation policies within the current SMC can then be used to

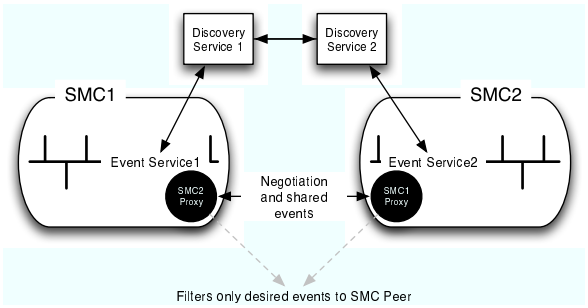

Figure 2.2: Self-Managed Cell Collaboration

determine which mission is loaded and instantiated onto the discovered SMC [6]. These collaboration methods at the system level can then be used in support of collaboration at the SMC application level. Policies, for example, from the system and management level can be used to determine application collaboration limitations and the extent of application discovery and advertisement.

Seamless collaboration at the application level, supported by lower-level collaboration architectures like the SMC, is difficult. It is unlikely that discovered SMC services and applications will adhere to a common language or naming structure. It is likely that different devices and applications will originate from different vendors who use different semantic descriptions. Alternatively, semantics are user-defined and thus subject to great variation [8]. Ontologies are used to solve the semantic difference problem between application and application content. An ontology is a shareable vocabulary written to permit common understanding and facilitate interaction between computer systems [11]. Ontologies capture knowledge of a given domain in a generic yet formal way, so that it can be reused and shared across applications and users. Ontologies are generally created via a man-made, time-consuming process where humans attempt to define all aspects of a system in a very explicit fashion. Frequently, different ontologies define very-similar knowledge. Mapping between ontologies associates terms defined in one ontology with terms in another. Currently, such mappings between ontologies are identified manually. This is extremely resource intensive, not always possible (particularly in ubiquitous environments like ours) and susceptible to ontology change. The Semantic Web serves to highlight an environment where automatic ontology mapping is a necessity [11]. Automatic ontology mapping, which removes the human element from the activity, covers a large number of fields from machine learning and formal theory to database schema and linguistics. Applications also range significantly, from academic prototypes to large scale industrial applications [12]. Most

systems are fairly complex, resource intensive creations and, as such, are not deployable in our resource-limited, ubiquitous computing environments [13–15].

Our automatic mapping mechanism described herein is deployable on an SMC at the application level and will permit us to carry out limited ontology generation and mapping between SMCs wishing to collaborate and share data.

Chapter 3

Scenarios

This chapter presents three motivating scenarios which aim to highlight potential uses for the work described in this dissertation. The scenarios described highlight a use, generally, for the SMC architecture and, more specifically, highlight a requirement for the application level mapping of data to be shared between peers.

Where appropriate, the scenarios are highlighted alongside real-world application examples which exhibit a particular need for the system, or where an emerging need has been identified. The goal of this chapter is to highlight a set of requirements and concerns for potential applications in ubiquitous environments, which should be addressed by the solution mapping mechanism described later.

The scenarios are listed from Section 3.1 to 3.3, where justification is also provided for their inclusion. The initial Healthcare scenario described merely highlights the motivation behind AMUSe and the SMC and will not be discussed in further detail. The SMC architecture in then used within the remaining 2 scenarios as an enabler to further explore application level ontology conflict and the development of suitable mapping mechanisms.

3.1 Healthcare Self-Managed Cell

An SMC may represent a patient body-sensor network consisting of a number of sensors and alarms. Implanted or wearable sensors capable of monitoring cardiac activity, insulin delivery, brain stimulation etc. continuously forward data to base stations. These support assessment, monitoring and treatment in everyday environments. A software infrastructure is required to allow such environments to become autonomous, managing their own evolution and configuration changes. Patient data is then made available remotely to health workers, enabling constant, accurate patient monitoring. This could provide doctors the data required to advise patients or forewarn emergency teams of danger etc. Such

data would be passed to such institutions via a form of e-Health Service Provider.

This example scenario is responsible for driving the development of the SMC architecture, as previously described in Section 2.2. The SMC architecture is then used as the hardware and software supporting environment for the ontology mapping worked described in this dissertation.

3.2 Music Player Composition

The principle domain of interest is that of peer-to-peer music sharing. The ability to see and listen to the music of others became prominent when Apple Inc. released a version of iTunes that supported the sharing of music collections on the same sub-network through the DAAP protocol [16]. Suddenly, individuals could listen to and examine not just their own music collections but that of any other connected user. Similarly, and more recently, Microsoft Inc. released the Zune portable media player [17]. A key differentiator between these two systems is Zune-to-Zune wi-fi communication, allowing sharing of songs, recordings, playlists and pictures with other Zunes. This change, from music players as a single-user jukebox application to a tool for music sharing, brings with it the potential for further study, particularly in relation to our SMC architecture and SMC application collaboration.

Music sharing has been realised in the ubiquitous and mobile computing world but with limitations. Semantic differences are very often visible in the music collections of peers, and this often proves a barrier to successful peer-to-peer collaboration and music sharing. Consider this example: Bobs discovers Alice's music collection whilst sitting on a train and wishes to listen to her music. He loves Indie music and searches Alice's music collection for this type of music. Disappointingly, nothing is returned from Bob's search as Alice has no genre defined in her library as "Indie", despite Alice having a wealth of tracks that Bob would commonly define as "Indie". There is a clear semantic difference in the way Bob and Alice classify their music collections; whilst this is a standard feature of personal music collections, overcoming these differences automatically would undoubtedly enhance the users' music sharing experience.

I will attempt to develop an SMC with a music player as an available service capable of utilising core SMC services such as the discovery and policy service to enhance peer-to-peer application interaction. The discovery service is responsible for initial SMC detection and the advertising of a music player as a cell service, and the policy service allows individuals to control access by others to their music together with rules for sharing etc. The music player running the SMC architecture could further benefit from an ontology mapping

mechanism which would reduce semantic differences between peer music collections and enhance the user music sharing experience.

3.3 Inter-Library Composition

In support of the mechanisms developed to solve the issues described in Section 3.2, this dissertation also investigates book collections held by a number of large libraries and attempts to overcome ontological difference that exist in the categorisation of these book collections. This work is intended as an additional scenario to extend the use of the mapping mechanism developed and provide an additional proof of concept.

Libraries are very similar to music collections in that they contain many objects (books) which are categorised based on aspects of meta-data (ISBN, Name, Subject, Author etc.) with differences frequently existing between libraries in the exact classification of particular books. Two libraries, for example, may contain the same book yet categorise it very differently according to Subject.

The issue of library categorisation disparity across collections is not a new problem and previous solutions have attempted to ensure that all libraries follow a set naming convention in relation to book meta-data aspects such as Name, Subject and Author, and then categorise books according to the titles laid down by a small number of leading institutions such as the American Library of Congress. This solution has worked to some extent in that many British and US academic and public libraries follow the convention, however it is very inflexible and does not support the situation where libraries opt not to follow the convention, particularly other international academic and public institutions.

In investigation I will attempt to confirm if, and to what extent, libraries categorise identical books differently and then investigate to what extent the SMC mapping mechanism will enable the dynamic generation of a common ontology which would overcome the semantic differences between local and international libraries.

Chapter 4

Related Work

This chapter surveys the major literature reviewed and provides a representative sample of the related work encountered. An abundance of material exists and to keep this chapter to a reasonable size, only an overview of the various areas is presented.

A preview of existing architectures and systems supporting ubiquitous computing environments is introduced in Section 4.1. Section 4.2 describes systems supporting applications collaboration with particular emphasis placed on the field of automatic ontology generation and mapping. The chapter concludes in Section 4.3 where related work in relation to the main dissertation focus, application sharing in the music sharing domain is described.

4.1 Ubiquitous Environments and the Self-Managed Systems

Work on policy-managed systems has been undertaken for many years aimed at the management of large-scale distributed systems but does not scale down to small devices suitable for ubiquitous computing environments. Such systems have included PDL [18], PMAC [19] and Ponder [10].

A number of pervasive/ubiquitous/multi-agent research projects are architecturally similar to the SMC. These include Aura [20], Ninja [21], CoBrA [22] and Gaia: Active Spaces [23].

The Ninja Project seeks to enable robust, distributed Internet services, and to permit extremely heterogeneous devices to seamlessly access these services. Similarly, Gaia: Active Spaces is a middleware operating system, that runs on top of popular operating systems, and can manage resources, devices and distributed objects in a room, building, or physical space. Context Broker uses a Context Broker Architecture (CoBrA) together with

OWL for modelling ontologies of context and for supporting context reasoning. PICO [19] is a middleware platform to enable effective communication and collaboration among heterogeneous hardware and software entities in pervasive computing. The PICO concept of a community is similar to the SMC, but the SMC focus is to facilitate self-configuration and self-management using policies. CodeBlue [24] is an ad-hoc sensor network infrastructure for emergency medical care. It integrates low-power, wireless sensors, PDAs and PC-class systems to provide a combined hardware and software platform for medical sensor networks. CodeBlue also provides protocols for device discovery, publish/subscribe, multi-hop routing and a simple data query interface for medical monitoring. CodeBlue investigates the data rates, node mobility, patterns of packet loss and route maintenance of the wireless sensor network, while the SMC framework focuses on the management of body-sensor networks using policies.

Work in the field is clearly vast although the SMC architecture can generally be differentiated from others in that existing systems do not tend to cater for interactions between ubiquitous environments and many existing systems tend to focus on the ubiquitous coverage of very defined spaces, such as the deployment of an office monitoring system, for example. The contribution with respect to these systems and techniques is a generic approach to ubiquitous systems management. The architectural pattern applies at different levels of scale and adaptation is controlled through policies. The SMC architecture provides truly autonomic management via configured policies and be scalable from managing simple personal-area networks as well as for large-scale network infrastructures and distributed applications.

4.2 Systems Supporting Application Collaboration

Automatic Ontology generation and mapping has seen a surge of research interest in recent years, perhaps due to the rise of the semantic web and the vast number of ontologies generated from.

Formal ontology generation approaches have modeled ontologies using graphs, logic and models with mappings being developed from viewing graph, logic and model convergence [13–15]. Current software systems that automatically generate ontology mappings are ONION [15], MAFRA [25] and IFF [23]. ONION generates mappings using graph transformations. MAFRA combines different similarity measures, both lexical and structural, to establish the mappings. IFF is based on convergence between logical theories. [12]

MAFRA, an Ontology Mapping Framework (MAFRA) for distributed ontologies in the Semantic Web, provides an approach and conceptual framework to give a generic

view onto the overall distributed mapping process. The system specifically focuses on the semantic mapping phase where a semantic bridge meta-ontology is instantiated when mapping between two domain ontologies. User participation in this mapping approach is fundamental and basically requires that each user determine how their ontology maps to the bridge meta-ontology. The bridge meta-ontology is then used to map between user ontologies. It is unlikely that this approach would be feasible in today's dynamic, ubiquitous environment. Users cannot be expected to determine their ontology's relationship to a bridge meta-ontology, particularly in a mobile environment or across a very large system such as a music collection, nor can they expect the availability of such a bridge. Similarly, ontology mapping as provided by IF-Map provides automated support in the alignment of ontologies by automatically generating mappings between a reference and local ontologies. Again, this work does not want to rely on a reference ontology but simply use the existing peer ontologies and their associated meta-data for the mappings.

ONION transforms source ontologies into graphs. The nodes and the edges of the graphs are then used to match graphs and therefore ontologies. Nodes are matched based on their names and a set of user-defined synonyms. ONION uses similarity between concept names for mapping. Mappings work well for ontologies having specialised terminology like medical ontologies where each concept is a disease and each disease has a unique name. Their matching accuracy decreases when mapping ontologies with more general terminologies. The mapping of large peer collections, particularly collections subject to great variance and user definition, would undoubtedly prove problematic for ONION to map.

The ontology mapping mechanisms defined are unlikely to be suitable for use in the ubiquitous environments defined in the scenarios presented in Chapter 3. They have primarily been designed to provide automated administrative assistance when mapping well defined but conflicting ontologies in traditional conflicting environments. They require considerable user input and tend to focus on the use of a bridging ontology, a resource unlikely to be available in the ubiquitous world. Furthermore, the mapping mechanisms would likely struggle in the undefined and uncontrolled ubiquitous world. Most mechanisms are also not suitably lightweight so as to be deployable on resource limited devices

SMC collaboration will be pivotal to the research ensuring the ability of one self-managed environment to interact seamlessly with another, particularly at the application level. The use of automatic ontology mapping techniques can provide some of the enabling structured ontologies and facilitate increased levels of SMC collaboration through the mapping of ontological differences between SMC application level services. The primary

contribution here is using the correlation of meta-data values in distributed collections to construct context-specific shared ontologies. The integration capabilities of SMC's can be exploited for the negotiations that are required. Furthermore, the ontology mapping mechanism must be suitably lightweight so as deployable on resource limited devices, yet suitably powerful so as to be functional in ubiquitous environments.

4.3 Current Music Sharing Technologies

This Section will aim to highlight some of the emerging software and hardware technologies which would suggest that the music sharing scenario identified in Chapter 3 is indeed current and a relevant area within which to research and develop.

At present there are many new technologies and applications appearing on the market which aim to support and enhance the idea of automatic data sharing between peers. Such applications are particularly prevalent in the music sharing world where the popularity of personal music players such as the iPod and the Zune has driven the demand for applications which support and enhance media sharing between peers.

Hardware such as Apple's iPod and Microsoft's Zune are being complimented by applications such as tunA[1] and DotTunes[2], both of which aim to facilitate the wireless sharing of music between peers in a controlled and monitored fashion.

TunA is a wireless application that allows users to share music locally through handheld devices. Users can "tune in" to other nearby tunA music players and listen to what someone else is listening to. Developed on iPaqs, the application displays a list of people using tunA that are in range, gives access to their profile and playlist information, and enables synchronized peer-to-peer audio streaming. Similarly, with DotTunes users can share music in a controlled and secure environment. One can create user names and passwords which allow each user to listen only to the playlists which they are authorised. One can also ban or allow certain IP addresses to block unwanted visitors. SSL layer encryption can also be enabled to protect content and ensure the highest level of security.

Microsoft, via the Zune itself, are also supporting social music sharing and application collaboration. The Zunes tag line has always been "welcome to the social", and now Microsoft has rolled out its own social networking site for Zune owners, called "Zune Social". Zune users are able to create a customisable "Zune Card" that automatically updates to reflect the music they are listening to on their Zune or with Zune software on

[1] http://web.media.mit.edu/ stefan/hc/projects/tuna/
[2] http://www.dottunes.net/

their computer. Other members of Zune Social can play samples of the songs a user has been playing directly from a friends Zune Card. "Zune Social" also offers standard social networking features such as profiles, messaging, friend lists, and so on. Dedicated music-based social networks such as Last.fm[3], iLike[4] and Goombah[5] already integrate with the iPod/iTunes ecosystem, and there are continuous calls for Apple itself to enhance their social features on iTunes (patents have recently been filed by Apple in the face of such calls[6]).

iLike, perhaps being the most advanced peer music mapping mechanism currently available, uses clip-matching technology, selecting clips of songs in your library which are then examined by iLike servers, matched with music data collected from other users, and on occasion made available from the servers as audio samples for public playback. Similarly, Goombah scans one's iTunes library, finds people that like the same style of music, and makes recommendations from their collections.

4.4 Summary

Having studied the described applications and technologies, it is apparent that the SMC is a very suitable supporting architecture on which to develop a mechanism to facilitate the mapping of peer data, particularly music. Similarly, current applications which promote social music sharing, as described in Chapter 3, do not seem to cater for the ubiquitous environments in which such music devices are commonly located. Most of the methods described rely on third party applications, centrally controlled servers and complex recommendation engines that do not lend themselves well to the mobile world.

[3] http://www.last.fm/

[4] http://www.ilike.com/

[5] http://www.goombah.com/

[6] http://www.engadget.com/2007/07/12/apple-patents-method-for-iphones-and-ipods-to-chat-wirelessly/

Chapter 5

Design and Fundamental Concepts

This chapter justifies the design goals and fundamental concepts required to successfully enable application collaboration through the use of the meta-data which describes the data used within the applications. The chapter describes the fundamental design concepts with reference to the music sharing scenario as defined previously in Section 3.2. It should be noted that the design principles apply to all Scenarios defined in Chapter 3 and many further scenarios that comply with the restrictions of the mechanism defined in Section 5.2.

Section 5.1 highlights the research undertaken to identify the need for the mapping mechanism described herein. Section 5.2 describes the initial design of the mapping mechanism and Section 5.3 then describes in greater detail, the protocol defined.

5.1 The Requirements for a Mapping Mechanism

To confirm the need for meta-data mapping in the music sharing context, 17 user music collections were analysed comprising 64,704 songs. Analysis of Genre meta-data values for the same artist or even the same song across user collections reveals very interesting results. There were a total of 6,040 artists and 462 distinct music genres in the libraries studied. The existence of 462 distinct genres indicates immediately that there are going to be vast ontological differences between the music of only 17 peers. Apples iTunes, for example, only contains approximately 30 different default genres, indicating that user-defined genres are very popular. The analysis also highlighted that approximately one third of all artists had more than one genre associated with them across the 17 user libraries.

Table 5.1 shows 6 popular Artists from the libraries studied and the number of unique genres with which they were associated. The results clearly highlight a vast difference across peers in the meta-data values associated with their music tracks. This was apparent

Artist	Number of Unique Genres	Genres
Miles Davis	3	Alternative and Punk, Jazz , No Genre
Mozart	3	Classical, Classicism, Concerto
Marvin Gaye	4	Dance, Electronica, RandB, No Genre
Bob Dylan	6	Folk, Pop, Rock, Soundtrack, Various, No Genre
The Beatles	7	Alternative Rock, Dance, Electronica, Pop Rock, Rock and Pop, Rock and Roll, No Genre
Oasis	8	Alternative, Alternative and Punk, Alternative Rock Brit Pop, Pop, Punk, Rock, No Genre

Table 5.1: Genres Associated with Artists

for all track meta-data aspects, such as Track Size, Length, Album, Format and Artist.

Similar results were apparent when data from 10 different library book collections was collated and analysed. Requests were made to a large number of local and international libraries for a listing of all texts in the general classification, Computing/Computer Science (this subject choice was simply to limit the size of the book collections returned for analysis simplicity). The libraries analysed included the National University of Singapore[1], The British Council Library of India[2], The University of Melbourne[3] and 7 British academic institutions[4]. The library data received was primarily from academic institutions and in the English language, again both to increase analysis simplicity and to constrain variances in the testing environment.

Again, similar to the music sharing scenario, library book meta-data exhibits the necessary features to be supported by a possible mapping mechanism as described in Section 3.3. All libraries studied contained similar, if identical, meta-data aspects such as Title, Author, Subject etc. and variances were frequently identified in the way in which libraries categorised identical books by Subject.

Interestingly, the majority of libraries tested, even the international libraries, adopted the Library of Congress Subject Heading (LCSH) framework. The LCSH framework was originally developed to provide a controlled vocabulary for subject indexing the Library of Congress (LC) collection in the USA. It is now widely used by other libraries and indexing agencies and in online bibliographic databases. Subject headings are established as they are needed to catalogue materials being added to the LC collection. Terms in current use are selected in establishing new headings. Changes are made to maintain the currency and

[1] http://libpweb.nus.edu.sg

[2] http://library.britishcouncil.org.in/

[3] http://www.lib.unimelb.edu.au/

[4] The Universities of Chester, Glasgow, Hull, Liverpool, Paisley, St Andrews, Sheffield

viability of the terms, although decisions on changes also take into account the need for stability of terms. Such an approach would suggest that the results expected when testing Subject heading differences across libraries would be minimal. Although the variances noted were not as significant as those in the music player example they were still frequent.

All libraries analysed exhibited the necessary meta-data differences that would suggest that a suitable mapping mechanism would improve cross library collaboration and data sharing.

5.2 A Mapping Mechanism Overview

The design of a mapping mechanism is restricted to applications that manipulate data that conform to a common schema - i.e. the application expects to access a data collection that can be modelled as a relational table; each row of the table corresponds to one object (e.g. a musical track), and each column corresponds to a metadata attribute for that type of object (e.g. Genre, Artist); finally, one, additional column containing the value of the object is included in each row (e.g. the actual mp3 encoding of a musical track). While in normal usage one would expect the possible values for a particular meta-data attribute for an object to be a small, enumerated set, this is not required; any legal value for the data type of the attribute (e.g. String) is permissible; it is also permissible for any meta-data attribute cell to be empty, with the exception of the value.

Using the music sharing example, the collection of tracks used by a particular player can be represented as shown in Table 5.2.

Title	Artist	Composer	Genre	Album	Size(mb)	...	Value
Son	Jethro Tull	Ian Anderson	Rock	Benefit	2.77		mt000001.mp3
Black Hole Sun	SoundGarden	Chris Cornell	Grunge	Superunknown	5.02		mt000002.mp3
Exsultate, jubilate	Kiri Te Kanawa	Mozart	Classical		14.11		mt000003.mp3
Rusty Cage	Johnny Cash	Chris Cornell	Country	Unchained	1.31		mt000004.mp3
Hush	Tool		Metal	Opiate	1.30		mt000005.mp3
Sleeping	The Band		Country Rock	Stage Fright	3.11		mt000006.mp3
Hello	Evanescence		Gothic Rock	Fallen	3.48		mt000007.mp3
...							

Table 5.2: An Example Home Collection

Even though the discussion is dominated by music sharing examples, other types of data collections are accessed in this way; for example, the collection of books maintained by a library. As the results highlight in Section 5.1, the meta-data for multiple book libraries also shows similar disparities across Subject Headings. This library example shows that the need for dynamic ontology mapping is not restricted to the ubiquitous computing domain.

In such environments, the definition of the data within the collection is often of personal or institutional determination with the majority of collections having no clear single point of authority. For example, most libraries use different naming standards for cataloguing and categorising books, and there are multiple vendors who determine, differently, the classification of music tracks, not including individuals who choose to select a personal naming scheme. These differences become problematic when searching the data collection of a collaborating peer within an autonomous environment.

The application user typically navigates through the meta-data space defined by the data collection using a variety of techniques; each navigation query can be mapped to an equivalent SQL query on the data collection; if more than one object is returned as the result of a query, the user is then able to choose a particular object for manipulation.

Each user is associated with a "home" collection of objects; in the music sharing example, it is the collection associated with the user's music player; difficulty can ensue when the application has access to one or more "foreign" collections in addition to the "home" collection. The user is most familiar with navigation through the "home" collection; in order to effectively access objects in the "foreign" collections, it is important to map the meta-data values that describe the "foreign" objects into values that have meaning to the user. Absent an overarching constraint on the values that can be used for the meta-data attributes for objects in the collections, this mapping must be determined dynamically. The basic result of a successful mapping is to temporarily import the objects in the foreign collection into the home collection, with meta-data attribute values for the foreign objects mapped to the home collection ontology.

In general, the meta-data attributes exhibit correlated values within a collection - i.e. many objects with $attribute_i = value_i$ also have $attribute_j = value_j$. The degree of correlation between $attribute_i$ and $attribute_j$ will depend upon: the attributes chosen, the nature of the collection, and the degree of consistency in value assignment when objects are added to the collection. For example, most artists are strongly correlated with a particular genre (e.g. all tracks produced by Pearl Jam are associated with the Grunge genre), while release dates are only weakly correlated with a particular genre (e.g. Grunge is correlated with release dates 1990 and beyond, but not before). Such correlations can be asymmetric due to the fact that some attributes have broader scope than others; the correlation strength is a measure of the predictive power of one value over the value of the other (e.g. Pearl Jam strongly predicts Grunge, but Grunge predicts Pearl Jam, Soundgarden, Alice in Chains, etc.). Figure 5.1 identifies the meta-data attributes which exhibit the greatest predictive powers across 17 user collections for nine standard meta-data aspects commonly

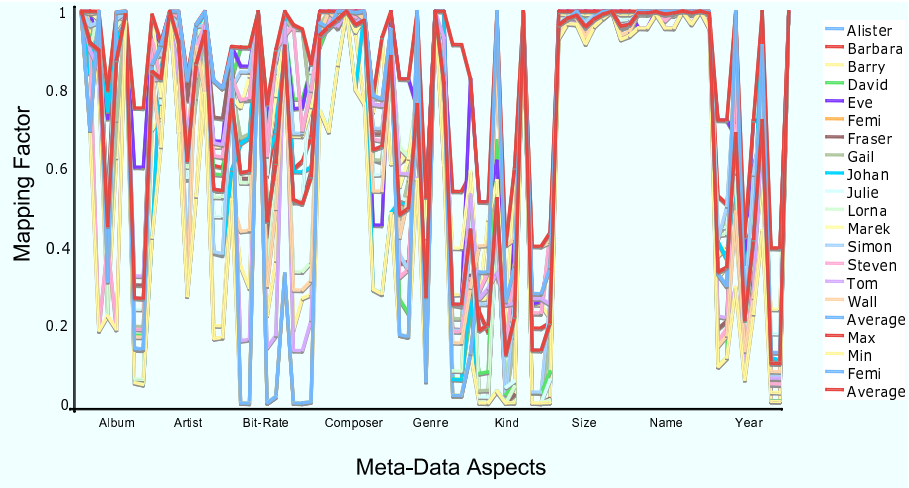

Figure 5.1: Average Predictive Power of Meta-Data Aspects across Music Collections

found in music collections. The attributes were analysed to determine how well each of their meta-data attributes could determine the contents within them. Aspects such as Name and Genre show a consistently high Predictive Power meaning they are attributes values suitable to use for mapping of music tracks. The Predictive Power is explained in further detail below.

Consider a collection of N objects, and each object has M meta-data attributes associated with it. Let us focus upon two attributes, i and j. In a particular collection, $Attr_i$ takes on values $v_{i1}...v_{in}$; similarly, $Attr_j$ takes on values $v_{j1}...v_{jm}$. All of the tracks in the matrix can then be analysed to yield the following matrix (Table 5.3):

$Attr_i/Attr_j$	V_{j1}	V_{j2}	...	V_{jn}
V_{i1}	C_{11}	C_{12}		C_{1n}
V_{i2}	C_{21}	C_{22}		C_{2n}
...				
V_{im}	C_{m1}	C_{m2}		C_{mn}

Table 5.3: Pairwise Classification of Objects in a Collection

where C_{k1} is the number of objects in the collection that have $Attr_i = V_{ik}$ and $Attr_j = V_{jl}$. It is informative to consider two limiting cases:

1. $Attr_i$ is strongly correlated with $Attr_j$: in this case, if there are N_{ik} objects with $Attr_i = V_{ik}$, then most of those objects will have $Attr_j = V_{jl}$ for some l; note that

by definition, $N_{ik} > 0$.

2. $Attr_i$ is not correlated with $Attr_j$: in this case, the N_{jk} objects with $Attr_i = V_{i,k}$ are distributed over many different values for $Attr_j$.

Summing over the pairwise matrix in Table 5.3 determines the predictive power of $Attr_i$ for $Attr_j$ as well as the predictive power of $Attr_j$ for $Attr_i$. One such formulation is as follows:

$$\text{predictive power}_{i,j} = \sum_{k=1}^{m} \frac{max_l\{c_{kl}\}}{\sum_{l=1}^{n} c_{kl}} \tag{5.1}$$

Obviously, the predictive $power_{j,i}$ simply requires that k is swapped for l and m for n in Equation (5.1). Performing this analysis for all pairs of attributes yields a correlation matrix of the form shown in Table 5.4. The value in the i, j^{th} cell indicates how strongly correlated values of $Attr_i$ are to values of $Attr_j$; obviously, the diagonal elements have a value of 1. Armed with this correlation information for the home collection, a protocol that uses this mechanism to dynamically map objects from a foreign collection into the home object ontology is described.

	$Attr_1$	$Attr_2$	$Attr_3$	...	$Attr_M$
$Attr_1$	1.000	0.357	0.771		0.467
$Attr_2$	0.953	1.000	0.849		0.121
$Attr_3$	0.642	0.838	1.000		0.368
...				1.000	
$Attr_M$	0.125	0.294	0.186		1.000

Table 5.4: Predictive Power

5.3 The Basic Mapping Protocol

The general protocol is as follows: if one is interested in objects in the foreign collection with $Attr_i = Value_j$, and none exist, then one searches the i^{th} column of Table 5.4 from the home collection for the $Attr_j$ with the largest correlation value (excluding row i). One can then query for objects corresponding to known $Value_j$'s, and discover the $Value_i$'s that the foreign collection associates with those objects. One can then import objects with those particular $Value_i$'s, replacing the actual $Value_i$ with the value used by the home collection.

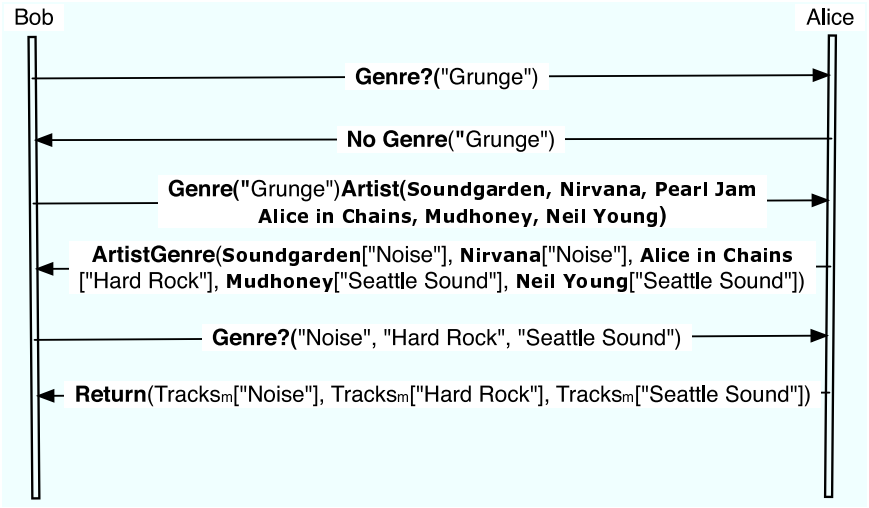

Figure 5.2: Music Mapping Sequence Diagram

To use these techniques in constrained ubiquitous computing environments, the mapping mechanism must be suitably simple and lightweight. A simple protocol has been developed to facilitate the sharing of music tracks between music players when meta-data differences exist between the collections. Two possible refinements to the protocol are discussed in Chapter 6 and the results of these refinements are presented in Chapter 7.

The most informative way to describe the basic mapping protocol is through an example in the music sharing scenario.

Figure 5.2 assumes that two peers are sitting on a train, each with a personal music player in the form of a PDA hosting a music streaming service; the two players have discovered each other, and the policies in the two players permit streaming of tracks from one player to the other. Once the players have bound together, the music services on each player can enter into the ontology mapping protocol shown in Figure 5.2. Bob's music service remotely performs a genre search on Alice's system for each value of the genre meta-data attribute defined for Bob's system; for example, suppose that one value of the genre attribute is "Grunge". Unfortunately Alice does not have any music defined as "Grunge", so the initial query returns a negative. The ontology mapping mechanism in Bob's music player selects a meta-data attribute strongly correlated with Genre, namely Artist, and queries Alice's player with a list of Artists associated with the genre "Grunge". Alice's music service then searches for those artists in her collection, and returns the most-prevalent genre value, if any, associated with the each artist in her collection. The protocol has established a Bob-specific mapping from his genre values to those used by Alice. Bob's

music service can now represent tracks in Alice's system using Bob-specific genre values. Of course, one expects the protocol to also be conducted in the reverse direction, to enable Alices system to establish an Alice-specific mapping from her genre values to those used by Bob. Besides enabling comfortable navigation over the other individuals collection and subsequent streaming, the mapping information can also be retained for future sharing with each other, or possibly to inform future negotiations with other peers. Note that other choices are possible for the last step of the protocol. The current protocol maps Bob's genre value to multiple genre values in Alice's collection. Another approach would be to only solicit the Alice genre value for the artist in Bob's collection with the largest number of tracks with that particular value, or the largest percentage of tracks with that particular value. The current approach maximises the number of tracks mapped to facilitate human navigation; more study is needed to determine if other approaches yield better results.

	Genre	**Artist**	**Name**	**Album**	**Year**	**BitRate**	**Kind**
Genre	1	0.579	0.25	0.57	0.475	0.646	0.885
Artist	0.818	1	0.623	0.861	0.855	0.865	0.921
Name	0.908	0.946	1	0.912	0.905	0.939	0.941
Album	0.857	0.893	0.275	1	0.793	0.888	0.964
Year	0.283	0.259	0.139	0.256	1	0.376	0.462
BitRate	0.238	0.188	0.187	0.234	0.184	1	0.939
Kind	0.18	0.13	0.039	0.035	0.064	0.299	1

Table 5.5: Predictive Power of Music Tracks

The mapping factor (attribute strongly linked to "Genre" in the preceding example) is determined through analysis of data collections of the appropriate type (music collections of tracks in the preceding example), as described in Section 5.3 above. Application of Equation (5.1) to the meta-data from 17 unique iTunes music libraries, comprising 64,704 tracks, yielded Table 5.5. For example, the mapping factors for music collections, shown in Table 5.5, indicate that there is a close relationship between Artist and Genre (0.818). In other words, if the Genre is not known then Artist is a good candidate meta-data attribute to map from, as is, Name and Album. Kind and Year, however, would not be suitable search attributes if the Genre was not known. Figure 5.3 highlights more specifically the attributes that have a strong mapping factor to Genre and other attributes which have a weak mapping factor.

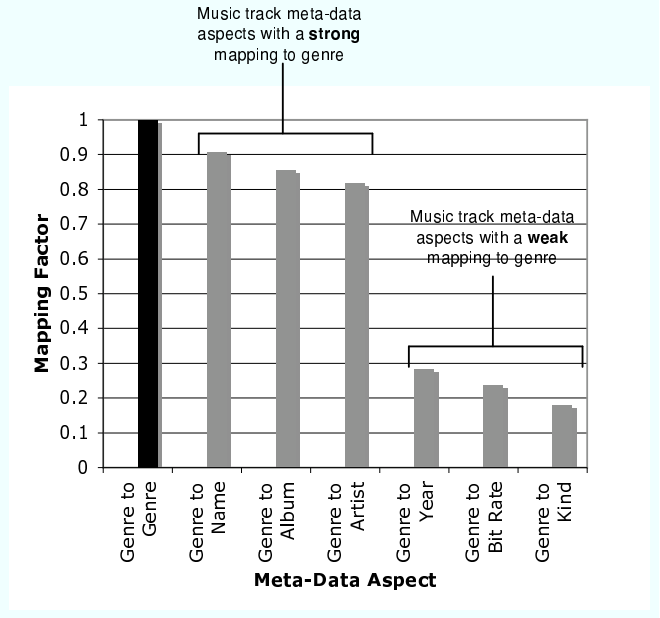

Figure 5.3: Mapping Factors to Genre

Chapter 6

Implementation

This Chapter describes the four main implementations that were undertaken to verify the design.

The Chapter begins by describing the primary implementation where the mapping mechanism described in Chapter 5 is implemented as part of a music sharing application running on the application layer of a Self-Managed Cell. A similar implementation is then described, again using the mapping mechanism described in Chapter 5, but this time verifying the mechanism in another environment, a traditional book library.

Finally two further implementations are described. In these implementations, the basic mapping mechanism itself is modified in an attempt to determine whether the quality, in terms of the relevance of the mapped music, or the quantity, in terms of the volume of the mapped music, is improved.

The summary of implementation results and an evaluation of each implementation described below is detailed in Chapter 7.

6.1 Music Collection Mapping and Self-Managed Cell

The Self-Managed Cell architecture running a music sharing service has been implemented as a test platform for the automatic data mapping technique. The music sharing service utilises core SMC services such as the discovery service and policy service.

The SMC has been built to run on a PDA (HP iPAQ hx4700, with a 624MHz XScale PXA270 processor and 64MB RAM, running Familiar Linux 0.8.4 or Windows Mobile 5.0). The SMC is written in Java, and uses JamVM 1.4.3 [26] in a bid to cut down on memory usage; a personal-area network holding a few event subscriptions and policies uses approximately 300kB of RAM, plus the size of the JVM and loaded classes. The policy service used is Ponder2 written in Java 1.4. In terms of cell connectivity, the SMC

architecture has been successfully implemented over WiFi (UDP/IP), Bluetooth, ZigBee and basic IEEE 802.15.4 (BSN nodes) [2].

The music player, built to run as a service on an SMC, is also written in Java 1.4. The player enables a user to search the music collection of other discovered music players and stream music found from such a search via wifi to the user's music player. It uses a the Digital Audio Access Protocol (DAAP) [27] which acts as an HTTP server for advertising and streaming requested songs to clients. This was introduced by Apple in iTunes v.4. The music player has been successfully tested under J2SE. The music player is approximately 4mB in size and has a memory footprint of around 15-30mB depending on activity status i.e. idle, playing, streaming etc.

The music player relies upon the mechanism documented in Section 2.3 for establishing the initial peer-to-peer binding between a pair of music players running as services on SMCs. For example, when Bob's music player SMC discovers Alice's music player SMC, Alice's SMC is assigned a role, of type MP3 player, in the domain structure. A role-dependent obligation policy is then used to specify which mission should be instantiated on the SMC of Alice. The SMC of Alice is likely to perform a similar function on Bob's music player. Policies used can include: a policy for checking music player availability on a peer SMC, a policy which starts the mapping mechanism if peer connection is established and a policy which defines the level of access a peer can have to your music collection. Upon completion of the binding, the availability of the music player service on each player is announced by an event in the other SMC, and each player can then engage the mapping mechanism with the other.

Mapping can be eager, meaning the entire libraries are mapped on initial peer connection, or lazy, in that mappings are only carried out when meta-data conflict is detected during use.

6.2 Library Collection Book Mapping

The basic mapping mechanism was used to compare the books contained in the libraries of a number of local and international book collections. The design of a mapping mechanism is well suited to being used for the comparison of library data - i.e. the application expects to access a data collection that can be modelled as a relational table; each row of the table corresponds to one object (e.g. a book), and each column corresponds to a meta-data attribute for that type of object (e.g. Author, Date, Subject);

A small Ruby[1] implementation was developed to identify similar books in a library collection where a Subject category available in the local collection, and used for search purposes, was not defined in the foreign collection and a mapping based on Subject was therefore produced. From testing, Subject was determined to have a strong correlation with Author and Title.

The basic mapping mechanism identified local meta-data from the meta-data aspects with the strongest correlation value (Author and Title) and mapped to a related Subject meta-data values in the foreign collection and returned the books within those Subject categories.

The technique proved particularly useful in mapping books to Subjects across libraries which did not follow the Library of Congress subject naming standards. This was mostly evident in the foreign libraries examined

6.3 A Variation on the Basis Mapping Mechanism

The basic mapping mechanism identifies an aspect of meta-data which is most strongly correlated to the aspect of meta-data being queried within the local collection and performs a mapping based on each item of meta-data within this aspect.

Testing highlighted that this approach was very thorough, in that it returned a very large number of related objects for each mapping. Testing, however, highlighted that many of the Objects returned were only loosely related and mappings were often over optimistic in terms of the number of matches returned.

A simple variation of the mapping approach was implemented and tested using the music sharing scenario. Instead of returning all Objects with meta-data matches to that of the correlated aspect, only the most strongly represented meta-data objects within the aspect identified are returned. For example, where an initial search of a foreign collection for “Indie” did not return any results, then the foreign collection is only searched using the item of meta-data appearing most frequently in the most correlated aspect of the local “Indie” genre. Previously a search of the foreign collection would have included all items of meta-data within the most correlated meta-data aspect.

This implementation was designed to reduce the number of results returned and increase the accuracy of the mapping.

[1] http://www.ruby-lang.org/

6.4 Mapping using Boyer-Moore String Matching

A string matching algorithm was integrated into the basic mapping mechanism to investigate the effects this would have on the results returned. The Boyer-Moore pattern matching algorithm [28] was effectively added to the basic mapping algorithm which would normally only permit exact, case-insensitive, string matches.

The Boyer-Moore algorithm is widely consider one of the most efficient string-matching algorithms in normal applications, for example, in text documents. The algorithm is optimal when the alphabet is moderately sized and the pattern is relatively long. The algorithm scans the characters of a word from right to left beginning with the rightmost character. During the testing of a possible placement of pattern P against text T, a mismatch of text character $T[i] = c$ with the corresponding pattern character $P[j]$ is handled as follows: if c is not contained anywhere in P, then shift the pattern P completely past $T[i]$. Otherwise, shift P until an occurrence of character c in P is aligned with $T[i]$. This technique avoids needless comparisons by significantly shifting pattern relative to text.[2]

This approach was an attempt to determine if the accuracy and size of the results returned would be improved. This mechanism would, in theory, identify related meta-data where, music tracks had only been partially recorded in terms of naming or where spelling errors perhaps effected parts of a track or artist name.

[2]http://www.personal.kent.edu/ rmuhamma/Algorithms/MyAlgorithms/StringMatch/boyerMoore.htm

Chapter 7

Measurement and Evaluation

This Chapter details some of the measurements obtained from each of the prototypes described in Chapter 6 and concludes by presenting a comparison of each mechanism.

The basic mapping mechanism described in Chapter 5 enables context-sensitive mappings to be produced when merging data collections. Provided individual collections are consistent in the manner that they assign meta-data attribute values, the mapping mechanism can provide some form of usable mapping.

The evaluation of the mapping mechanism focused on the mapping of music data collections, firstly due to the quality of the 17 test collections available, as described in Chapter 5.1, and secondly due to the availability of two independently composed databases of music categorisation which could be used determine the quality of the mapping between peers.

7.1 Meta-Data Attribute Values

The mapping mechanism described relies on the fact that there is an aspect of commonality in the meta-data attribute values which categorise the collection meta-data. This has previously been described as the peers having a partially shared ontology. It was therefore essential that the collections being suggested for mapping contained sufficient commonality in the meta-data attribute values.

The 17 music collections were initially evaluated to ensure that there was general consistency in the values that were assigned to the meta-data attribute values of each peer. In the case of the music collections over 95% of the meta-data attribute values were identical. This figure is very high, but clearly linked to the predominant use of iTunes for music collection management in the collections analysed. This also points to a general lack of user customisation in terms of these attributes. This supports the basic fundamental

assumptions of the mapping mechanism that peers will possess a partially shared ontology.

In light of the high iTunes use in the peer test set further analysis was undertaken to ensure meta-data attribute values, usually automatically created by music players, were similar/identical across music players and operating systems. Table 7.1 shows the most popular music players from Apple(iTunes[1]), Linux(Banshee[2]) and the Windows(WMP[3]) operating systems. As can be seen in the Table there is indeed commonality in the most significant meta-data attribute values across popular music players and one is therefore assured that the principle of a partially shared ontology is supported. Not all meta-data attributes available are shown in the Table, only those significant in terms of the mapping mechanism.

iTunes	WMP	Banshee
Album	Album	Album
Artist	Artist	Artist
Name	Title	Title
Genre	Genre	Genre
Year	Year	Year
Composer	Composer	-

Table 7.1: Meta-Data Attribute Values

Similarly, in relation to book library collections, meta-data attribute values tended to be very similar, if not identical across all libraries. The essential meta-data attribute values of peers such as Title, Author, Subject, Publisher, ISBN etc. tended to be identical for both local and international libraries.

It should be noted that it is not essential for all meta-data attribute names to be identical, but obviously at least one must match that in the peer collection. Preferably, that matching name would be one with a high predictive power. For example, it would be preferable if the meta-data attribute names “Genre” and “Artist” would be common across both collections.

7.2 Music Collection Mapping

The mapping mechanism, used to enhance collaboration between peer music libraries, has been fully tested and evaluated. Analysis of collaborations using the 17 peer music collections described in Section 5.1 revealed significant use of the mapping system, with song returns usually running into the hundreds where initial collaboration had revealed

[1]http://www.apple.com/itunes/, v6.0.1

[2]http://banshee-project.org/, v0.9.7

[3]http://www.microsoft.com/windows/windowsmedia, v9.0

few or no matches. 462 distinct genres were present across the 17 libraries, however, the majority of user's only used 30 genres each, highlighting a significant scope for mapping in relation to Genre.

The analysis of each mapping mechanism proposed took a standard form, and focused on the Genre-to-Artist mapping explained throughout this dissertation. The analysis took the following form: for all possible peer-to-peer mappings from the 17 test data collections Genres were mapped using Artist. Only Genres which were not held by the foreign peer, and would thus return no Object Tracks, were mapped. Tracks were returned from this mapping. AllMusic[4] and FreeDB[5] were used as oracles to determine if the tracks returned as part of the mapping were false positives or negatives. AllMusic is an expert compiled online music database and FreeDB is a database of music information compiled by the general public.

False positives describe tracks that should not have been returned during the mapping process and false negatives describe tracks which were present in the foreign collection that should have been identified and mapped by the mechanism but were omitted. Obviously, fewer false positives and negatives will represent a better mapping mechanism but this must be considered alongside the average number of tracks returned using the mechanism. The notion of false positives and false negatives are thus used to determine the accuracy of the Objects returned from the mapping in respect of the initial query. The comparison with an oracle, such as AllMusic and FreeDB, enabled the determination of the tracks which should have been returned if all collections conformed to that common ontology.

The results for the basic mapping mechanism are shown in Table 7.2. The results are shown based on an average of all tracks returned for each Genre-to-Artist mapping across all peers. The false positives and false negatives detected are also averaged across all collections.

Tracks	AllMusic		FreeDB	
Returned	False Positives	False Negatives	False Positives	False Negatives
723	495	19	380	39

Table 7.2: Genre Mapping Using the Basic Mechanism

The results can be explained as follows. The average number of tracks returned for a single genre mapping would seem to be very high. This is due to the fact that request for genres associated with artists known to be correlated with the home collection genre of interest genre request may result in the return of multiple genres from the foreign peer,

[4]http://www.allmusic.com/
[5]http://www.freedb.org/

and hence a large number of tracks. Multiple genre returns is the result of the mechanism requesting the genre associated with every artist in the genre they failed to locate in the foreign collection (see Chapter 5). AllMusic has a higher number of false positives in comparison to FreeDB due to the nature of the databases. FreeDB may contain multiple genres for every artist as a consequence of its public creation whilst AllMusic is far more strictly controlled, has a rigid categorisation structure and is expertly compiled. False negatives are similarly affected.

As an additional example, the results of a single Genre-to-Artist mapping results from a peer-to-peer collaboration are shown in Table 7.3. Only genre searches where no song results were initially returned are shown. Similar results were apparent across all peer mappings when using a meta-data aspect with a high Predictive Power.

Peer 1	**Peer 2 Returns after Mapping**		
Genre Request	Genres	Artists	Songs
Blues	2	337	3594
Classic Rock	2	282	2352
Electronica	1	115	587
Folk	2	282	2352
Rock/Pop	2	337	3594
Soul	1	11	109
Top 40	1	40	467

Table 7.3: Genre-Artist Mapping

The mapping results highlighted that the mechanism was able to successfully map ontological differences in the meta-data used in defining music tracks. The analysis did however reveal that although the mapping mechanism was capable of returning a large number of "related" tracks as a result of a peer mapping, many of the tracks tended only to be loosely related to the initial search query and the users' requirements.

The results revealed that on average approximately 40% of the tracks returned using the basic mapping mechanism were accurate when compared with the suggested track meta-data in each of the online music databases. For example, if a mapping determined that "Oasis" were in the "Rock" category, then the mapping would be deemed to perfect, as both AllMusic and FreeDB categorise "Oasis" as being a "Rock" band. The results do, however, very much depend of the quality of the individual's music collection meta-data, the approach they take to categorising their collection and the particular Peer Collection they are mapping with.

7.3 Library Collection Book Mapping

This section analyses the mapping mechanism when used to facilitate collaboration between library book collections. These collections, as with the music collections, revealed conflicting meta-data and was designed to highlight the applicability of the basic mapping mechanism described herein across all applications that manipulate data that conform to a partially shared ontology - i.e. the application expects to access a data collection that can be modelled as a relational table, as described in Chapter 5.

Initial analysis of the Collections received from the various National and International Libraries showed some disparity between meta-data values for identical books. For example, the Subject of the title "Java in a Nutshell" took over 5 values from the 10 libraries investigated (see Section 5.1).

Table 7.4 shows an example of a mapping of Library Subjects to Books where the initial Subject requested is not located in the foreign library. Local Books within the Subject requested are then used to complete the mapping.

Library 1	**Books Returned After Mapping**	
Request	Subjects	Books
Computing	3	1693
Standerdisation	1	436
Electronics	2	649
Java	1	297

Table 7.4: Subject-to-Book Mapping

Without an easily accessible oracle such as AllMusic and FreeDB, described in Section 7.2, it was not possible to determine the quality of the results using the false positive and false negative approach. However, it should be noted that the mapping of Library Book Collections would appear to exhibit similar results to those discovered when mapping Music Collections.

7.4 Mapping Mechanism Variation

This implementation was evaluated in the music sharing domain. In response to the low accuracy rates achieved in Section 7.2 above, and the high number of tracks returned, a slight modification was made to the basic mapping mechanism in an attempt to increase the accuracy of the tracks returned. The exact modifications made are detailed in Section 6.3. An example of a simple Peer-to-Peer mapping is shown in Table 7.5.

Table 7.6 shows the results for the basic mapping mechanism for the same data set used in Section 7.2. Again the results are shown based on an average of all tracks returned

Peer 1	Peer 2 Returns after Mapping		
Genre Request	Genres	Artists	Songs
Blues	1	269	2753
Classic Rock	1	101	984
Electronica	1	115	587
Folk	1	101	984
Rock/Pop	1	269	2753
Soul	1	11	109
Top 40	1	40	467

Table 7.5: Genre-Artist Mapping (Variation to Basic Method)

for each Genre-to-Artist mapping across all peers. The false positives and false negatives detected are also averaged across all collections.

Tracks	AllMusic		FreeDB	
Returned	False Positives	False Negatives	False Positives	False Negatives
458	238	33	168	73

Table 7.6: Genre Mapping Using the Mapping Mechanism Variation

Analysis of the variation to the basis mapping mechanism showed a slight increase in accuracy of the tracks returned (49-56%) and a noticeable decrease in the total number of tracks returned. The reduction in the number of tracks returned is due to limiting the mapped elements of meta-data to a single item deemed the Object most representative the set. This has also decreased the number of false positives as less tracks are now within the scope of being mapped. The false negatives increased as the reduction in the set size means related tracks have been omitted from the mapping.

7.5 Mapping using Boyer-Moore String Matching

Analysis of the addition of the Boyer-String Matching Algorithm to the basic mapping mechanism proved to be predictable (see Table 7.7). The number of results returned increased across all collections tested and the accuracy of the results decreased slightly from the basic mechanism results described in 7.2.

These results are predictable as the mapping mechanism will also detect like words, for example, “Oasis” and “Noasis” and identify them as being the same. It is also possible to identify a track as being a “match” where it has only a single identical string from a group of strings forming a track name.

There may be other possible methods of using the Boyer-Moore algorithm which could in-fact help increase accuracy.

Tracks Returned	AllMusic		FreeDB	
	False Positives	False Negatives	False Positives	False Negatives
901	570	24	412	30

Table 7.7: Genre Mapping Using Boyer-Moore

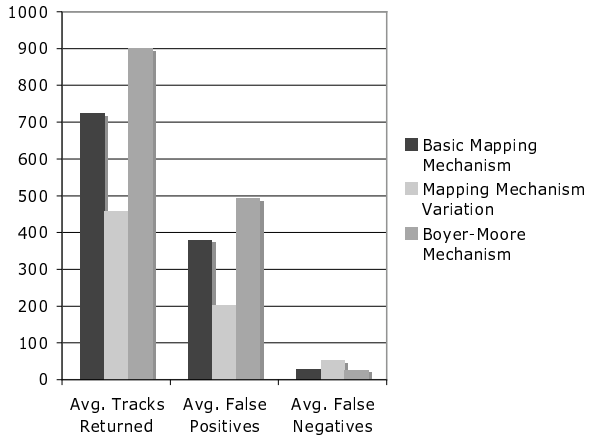

Figure 7.1: Comparison of Mapping Mechanisms

7.6 Mapping Mechanism Performance Comparison

Figure 7.1 shows a comparison of false positives, false negatives and average tracks returned for a Genre-to-Artist music collection mapping. The results for each mechanism would appear intuitive in that there is clearly a loss of accuracy when the basic mapping mechanism is selected and in return the number of tracks returned is increased. Likewise, the variation of the basic mechanism increases accuracy at a cost of the number of tracks returned.

The results would appear inconclusive in terms of determining exactly which mapping mechanism is optimal and it is likely that this will be dependant on user needs. It may be possible to allow the user to determine their mapping mechanism of choice depending on their situation.

Section 8.5 also highlights some additional variations to the mechanism which may be possible although these are retained for future work. Such variations include the use of multiple meta-data aspects in the mapping mechanism to determine mapping criteria and the use of Collection statistics to reveal the most predictive meta-data items to use when mapping. It is likely that the mapping accuracy will increase by further optimisation to the mapping mechanism.

Chapter 8

Discussions and Conclusions

8.1 Validation of the Thesis Statement

The Thesis Statement in Chapter 1 presented a number of assertions; those being,

- It is possible for independent, ubiquitous systems to dynamically collaborate at the application level
- It is possible to overcome semantic and ontological differences between systems at the application level, specifically applications defined by data collections characterised by meta-data.
- It is possible to do so when the applications possess only a partially shared ontology.

This dissertation has validated these assertions through the development of a automated ontology mapping mechanism that facilitates music sharing where the music meta-data for identical or similar tracks differs between peers. This system has been developed to run on the applications layer of an ubiquitous computing architecture, the SMC, and requires only that peers have a partially shared ontology, i.e. common aspects of meta-data.

8.2 The Approach to Application Collaboration

A novel automated application meta-data mapping mechanism has been described that supports application-level integration within ubiquitous systems. The mechanism facilitates the successful collaboration of data collections by using meta-data contained within the collections to identify areas of commonality between collections. The commonality identified is then used to automatically generate a common ontology and map between

the areas of conflict. By using associated meta-data information stored with music tracks, for example, the mapping mechanism was successfully able to share music between peers despite there being minimal commonality to support collaboration. The techniques used help establish the common ontology between peers which can be temporary or retained for future sharing with the connected peer. The system is suitably lightweight and resource efficient such that it can operate in constrained environments such as PDAs and mobile telephones.

8.3 The Self-Managed Cell

A Self-Managed Cell is a suitable host architecture for the application level integration mechanism. SMCs can support peer-to-peer collaboration through the provision of basic abstractions and protocols. Peers can exchanges management events, policies and data which can then be used as the backbone to facilitate successful SMC application interaction.

8.4 Implementation and Evaluation Method

A prototype music sharing application, existing as a Self-Managed Cell service, has been built and demonstrated on a full JVM under J2SE. A second exemplar implementation based on mapping between several traditional library book collections using a scaled up SMC architecture has been developed to highlight additional scope for the mapping mechanism defined.

Experimentation has also taken place with the basic mapping mechanism, described in Chapter 5. A refinement was made to the basic mechanism in which a pattern matching algorithm was used.

8.5 Avenues for Future Work

The mapping mechanism can benefit from a number of future improvements. At present the mechanism's simplicity can lead to large volumes of data, only vaguely similar, being returned after mapping, particularly in the instance of music sharing between peers with non-rigid and inconsistent music structuring methods.

It may be possible to enhance mapping quality by combining multiple meta-data attributes with strong mapping factors, such as artist and track name, or using other meta-data values from a collection such as most frequently or recently played track, in the

instance of music sharing, to enhance the mapping relevance. Similarly, the mapping factors, shown in Table 5.5, could be regenerated over-time as the music player collaborates with other peer systems.

The investigation and implementation of a lightweight version of the more sophisticated approaches to automatic ontology mapping which are beginning to appear [12] may also be possible.

There may also be aspects of the approach to SMC level integration management that can take advantage of the automatic ontology mapping mechanism, particularly in relation to semantic differences in the policies of peers.

8.6 Contribution

The primary contribution here is using the correlation of meta-data values in distributed collections to construct context-specific shared ontologies. The integration capabilities of SMC's can be exploited for the negotiations that are required. Furthermore, the ontology mapping mechanism is suitably lightweight so as to deployable on resource-limited devices, yet suitably powerful so as to be functional in ubiquitous environments.

A further advantage of the mapping mechanism described is the need only for a partially shared ontology. Peers only require a basic partially shared ontology, described herein as the aspects of meta-data. Such aspects can be described as the simple structures used to define collection meta-data such as Album, Subject, Artist, Name etc. Research identified that there were always at least a small common subset of these for every collection category identified.

Current applications and technologies do not facilitate the mapping of peer data in a distributed environment, where lightweight, non-intrusive approaches are required. Similarly, current applications which promote social music sharing, as described in Chapter 3, do not cater for the ubiquitous environments in which such music devices are commonly located. Most of the methods described rely on third party applications, centrally controlled servers and complex recommendation engines that do not lend themselves well to the mobile world.

Bibliography

[1] Joe Sventek, Nagwa Badr, Naranker Dulay, Steven Heeps, Emil Lupu, and Morris Sloman. Self-managed cells and their federation. In *CAiSE Workshops*, volume 2, pages 97–107, 2005.

[2] S. Strowes, N. Badr, N. Dulay, S. Heeps, E. Lupu, M. Sloman, and J. Sventek. An event service supporting autonomic management of ubiquitous systems for e-health. In *5th International Workshop on Distributed Event-Based Systems*, 2006.

[3] Sye Loong Keoh, Kevin Twidle, Nathaniel Pryce, Alberto E. Schaeffer-Filho, Emil Lupu, Naranker Dulay, Morris Sloman, Steven Heeps, Stephen Strowes, Joe Sventek, and Eleftheria Katsiri. Forthcomming: Policy-based management for body-sensor networks. In *4th International Workshop on Wearable and Implantable Body Sensor Networks*, 2007.

[4] N. Dulay, S. Heeps, E. Lupu, R. Mathur, O. Sharma, M. Sloman, and J. Sventek. Amuse: Autonomic management of ubiquitous e-health systems. In *Proceedings of the UK e-Science All Hands Meeting*, UK, 2005.

[5] E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, K. Twidle, S. L. Keoh, and A. E. SchaefferFilho. Amuse: autonomic management of ubiquitous systems for e-health. *Special Issues of the Journal of Concurrency and Computation: Practice and Experience*, 2006.

[6] A.E Schaeffer-Filho, E. Lupu, N. Dulay, S.L. Keoh, K. Twidle, M. Sloman, S. Heeps, S. Strowes, and J. Sventek. Supporting interactions between self-managed cells. *Submitted to International Conference on Self-Adaptive and Self-Organizing Systems*, 2007.

[7] S.Heeps, J.Sventek, N.Dulay, E.Lupu, A. E. Schaeffer-Filho, M.Sloman, and S.Strowes. Dynamic ontology mapping for interacting autonomous systems. In

Springer, editor, *Workshop on Self-Organizing Systems*, volume 4725, pages 255–263, 2007.

[8] S.Heeps, N.Dulay, E.Lupu, A. E. Schaeffer-Filho, M.Sloman, S.Strowes, and J.Sventek. The autonomic management of ubiquitous systems meets the semantic web. In *The Second International Workshop on Semantic Web Technology For Ubiquitous and Mobile Applications*, 2006.

[9] J. Kephart. The vision of autonomic computing. *IEEE Computing*, pages 41–50, 2003.

[10] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In *In Proceedings of the International Worksop on Policies for Distributed Systems and Networks*, 2001.

[11] G. Cañadas, M. Fernández-López, R. García-García, M. Lama, A. Sánchez-Alberca, and COS. Sorzano. Framework for automatic generation of ontology. *Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento*, 2004.

[12] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[13] A. Maedche, B. Motik, N. Silva, and R. Volz. A mapping framework for distributed ontologies. In *13th International Conference on Knowledge Engineering and Knowledge Management*, 2002.

[14] J.Y. Park, J.H. Gennari, and M.A. Musen. Mappings for reuse in knowledge-based systems. In *11th International Workshop on Knowledge Acquisition, Modeling and Management*, 1998.

[15] P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *7th International Conference on Extending Database Technology*, 2000.

[16] Apple. ipod and itunes. *http://www.apple.com/itunes*, 2007.

[17] Microsoft. Zune. *http://www.zune.net*, 2007.

[18] J. Lobo, R. Bhatia, and S. Naqvi. A policy description language. In *In Proceedings of the 16th National Conference on Artificial Intelligence*, USA, 1999.

[19] M. Kumar, B.A. Shirazi, S.K. Das, B.Y. Sung, D. Levine, and M. Singhal. Pico: A middleware framework for pervasive computing. *IEEE Pervasive Computing*, 2(3):72–79, 2003.

[20] D. Garlan, D.P. Siewiorek, A. Smailagic, and P. Steenkiste. Aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing*, 1(2):22–31, 2002.

[21] S.D. Gribble, M. Welsh, R. von Behren, E.A. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R.H. Katz, Z.M. Mao, S. Ross, and B. Zhao. The ninja architecture for robust internet-scale systems and services. *IEEE Computer Networks*, 35(4):473–497, 2001.

[22] H. Chen, T. Finin, and A. Joshi. Semantic web in the context broker architecture. In *IEEE International Conference on Pervasive Computing and Communications*, 2004.

[23] M Romn, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. Gaia:a middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, pages 74–83, 2002.

[24] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[25] Y. Kalfoglou and M. Schorlemmer. If-map: an ontology mapping method based on information flow theory. *Journal on Data Semantics*, pages 98–127, 2003.

[26] Robert Lougher. Jamvm. *http://jamvm.sourceforge.net/*, 2007.

[27] Chris Boot. Digital audio access protocol. *http://daap.source forge.net/*, 2007.

[28] R. Boyer and J. Moore. A fast string searching algorithm. *Comm ACM*, 20, 1977.